University of Alberta

Dielectrophoresis-Based Microfluidic Devices Fabricated by Soft Lithography

by

José Luis García Cordero ©

A thesis submitted to the Faculty of Graduate Studies
and Research in partial fulfillment of the requirements
for the degree of

Master of Science

in

Department of Electrical and Computer Engineering,

Edmonton, Alberta

Fall 2004

Library and
Archives Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-612-95749-7
Our file Notre référence
ISBN: 0-612-95749-7

The author has granted a non-exclusive license allowing the Library and Archives Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou aturement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Para mis padres es este pequeño fruto. Todo lo bueno que se produce en estas líneas ha sido en su mayor parte resultado de sus enseñanzas, bondad, cariño, confianza, y apoyo en todo momento.

Acknowledgments

This thesis is not only the effort of two years of work. Many people have contributed in different ways through 25 years with advice and support, either inside academia or outside of it. I would like firstly to express my gratitude for the financial support received from the people of Mexico through CONACYT (National Science Council for Science and Technology). My thanks goes to Dr. Backhouse for letting me join his group, the financial support I received from him during the last year of my studies, and the patience and encouragement to complete this project. Most of this thesis is the result of his comments and suggestions. I am truly indebted to him for having introduced me to the marvelous world of bio- and nanotechnology.

People in the Backhouse Lab created a good atmosphere and many laughs during my stay: Holly, Jay, Yao, Rubin, Alex, Ranjit, Vincent, Govind, Golnaz, Guijun, Dammika, Eric, and Kirat. Loi Hua always showed great disposition to help me out in everything he could, especially with electronics and the layout of the PCB's. Marwan Saab took the challenge to program the custom function generator this summer. I could not have fabricated any device if it was not for the training and advice provide by the Nanofab Staff: Stephanie Bozic, Aruna Kroetch, Keith Franklin, and Shane McColman. My sincere gratitude and millions of thanks to this group of people.

I also feel really indebted with the many teachings I have received from my friends, siblings, or acquaintances that in certain way have had an impact in my life. This is a good opportunity to pay my respect and tribute to all of them. Diana and Erick Garcia, Jose Ernesto Cepeda and Marco Antonio Mucino, have proven to be confidents and friends during my life. Professors Kande Mutsaku, Salvador Venegas, Neil Gress, and Rafael Garcia, give the encouragement (through valuable discussions and teachings) that I was looking for to continue studying. People from Monterrey and from everywhere for the good times I had with them: Rodrigo, Willy, Elias, Igmar, Francisco, Luis, Emmanuel, El Benjas, Tzipeti, David, Manus, Gerardo, my cousins, uncles, aunts, and forgive me if I am missing someone.

Living in Edmonton was a wonderful experience. The Mexican gang did not make me miss home; especially Dr. Concha who has been an excellent friend and roommate and Francisco Jimenez and Jesus Hernandez for the many talks and friendship. Last but not least, my heart and my love to a wonderful person, Susanne Marie Hutzal.

Table of Contents

1 **INTRODUCTION** 1
  1.1 Cell Separation 1
    1.1.1 Electrophoresis 2
    1.1.2 Flow-Field Flow Fractionation 2
    1.1.3 Filtration 2
    1.1.4 Fluorescence Activated Cell Sorters 3
    1.1.5 Magnetic Activated Cell Sorter 3
    1.1.6 Dielectrophoresis 4
  1.2 Micro-Total Analyses Systems 5
  1.3 Materials for Micro-Total Analyses Systems Technologies 6
    1.3.1 Silicon and Glass 6
    1.3.2 Polymers 6
    1.3.3 PDMS 7
  1.4 Core Components of Micro Total Analysis Systems 8
    1.4.1 Microfluidic Pumping 8
      1.4.1.1 Macro- and Micro- Pumps 8
      1.4.1.2 Electrokinetic Pumps 9
      1.4.1.3 Induction Electrohydrodynamic Pumps 10
    1.4.2 Microfluidic Valves 10
      1.4.2.1 Freeze-thaw Valves 10
      1.4.2.2 Micromechanical valves 10
      1.4.2.3 Hydrogel and Surface Tension Valves 11
      1.4.2.4 Multilayer Soft Lithography Microvalves and Micropumps 11
  1.5 DEP Devices Implemented on PDMS 12
  1.6 Scope of the Thesis 12

2 **THEORETICAL BACKGROUND** 14
  2.1 Dielectrophoresis 14
    2.1.1 Homogeneous Sphere in an Non Uniform Electrostatic Field 14
    2.1.2 Homogeneous Sphere in an Alternating Non Uniform Electric Field. 17
    2.1.3 Heterogeneous Spheres (Cells) 20
  2.2 Secondary Effects 24
    2.2.1 Pearl Chain Formations 24

3 **DESIGN OF DEP DEVICES** 25
3.1 Introduction 25
3.2 DEP Configurations 25
3.2.1 Polynomial Configuration 26
3.2.2 Quadrupole Dielectrophoretic Traps 27
3.2.3 Travelling Wave Dielectrophoresis 28
3.2.4 Interdigitated Electrodes 29
3.3 Levitation Effects 31
3.4 Considerations for Electrode Sizes 31
3.5 Simulations 31

4 **FABRICATION** 33
4.1 PDMS Master Mold Fabrication 33
4.1.1 Interconnections 34
4.1.2 PDMS Replicas 35
4.1.3 Discussion 38
4.2 Electrode Fabrication 38
4.3 Bonding 39
4.3.1 Introduction 39
4.3.2 Procedure 39
4.3.3 Alignment 40

5 **TESTING OF DEP DEVICES** 42
5.1 Effects of PDMS Channels 42
5.1.1 Experimental 42
5.1.2 Treatment of Fluidic Channels 44
5.2 Experimental Setup 45
5.3 Materials and Methods 46
5.3.1 Beads 46
5.3.2 Buffer Solutions 47
5.3.3 Cleaning Procedure 47
5.3.4 Channel Conditioning 47
5.4 Quadrupole Electrode Structure 47
5.4.1 Simulation 47
5.4.2 Results for White Sulfate Beads 48
5.4.3 Results for Surfactant Free Red Carboxylate-Modified Latex Beads 50
5.5 Polynomial Electrode Structure 52
5.5.1 Simulation 52
5.5.2 Results for Yeast Cells 53
5.6 Interdigitated Electrodes 56
5.6.1 Simulations 56
5.6.2 Separation of White and Red Latex Beads 57
5.7 Castellated Electrodes 59
5.7.1 Simulations 59

5.7.2 Results for White Sulfate Beads on Electrodes with Gaps of 80 μm 60
5.7.3 Results for Red CML Latex Beads on Electrodes with Gaps of 60 μm 61
5.7.4 Results II for Red CML Latex Beads 62
5.7.5 Separation of White and Red Latex Beads 64

6 **HARDWARE AND SOFTWARE TOOLS** 67
6.1 Function Signal Generator 67
6.1.1 Results 73
6.2 Software 75
6.2.1 Close Captioning 76

7 **FABRICATION OF MICROVALVES AND MICRO-PERISTALTIC PUMPS** 78
7.1 Introduction 78
7.2 Multilayer Soft Lithography 78
7.2.1 Multilayer Soft Lithography Valves and Pumps 79
7.2.2 Push-up and Push-down Valve Configuration 79
7.3 Fabrication of Microvalves and Peristaltic Micropumps 80
7.3.1 Flow and Control Layer Master Mold Fabrication 81
7.3.2 PDMS Casting 83
7.4 Design of a Setup to Control Microvalves and Micropumps 86
7.4.1 Pneumatic System 87
7.4.2 Electronic Valve Control 88

8 **CONCLUSIONS AND FUTURE WORK** 90
8.1 Summary 90
8.2 Future Directions 90

BIBLIOGRAPHY 95

APPENDIX A: PROTOCOL FOR CREATING A SOLUTION OF BSA COATED BEADS 105

APPENDIX B: PARTS LISTS 107

APPENDIX C: ELECTRODE AND PDMS MASKS 110

APPENDIX D: PRINTED CIRCUITS BOARD LAYOUT 112

APPENDIX E: SOFTWARE PROGRAMS: MICROCONTROLLER 114

APPENDIX F: SOFTWARE TOOLS 118

APPENDIX G: SOFTWARE CODE FOR THE CONTROL OF THE SOLENOID VALVES 144

APPENDIX H: MATLAB CODE CREATED BY FEMLAB 145

APPENDIX I: LIST OF SUPPLEMENTARY MATERIAL 153

List of Tables

Table 1. Physical and Chemical Properties of PDMS [43 – 45, 49 - 51] 8

List of Figures

Figure 1. A dielectric object subject to the presence of an inhomogeneous electric field experiences a net force to regions of high strength electric field. 16

Figure 2. The graph on the left shows the theoretical frequency dependence (complex permittivity, see eq. 17) of particles suspended in a medium that does not exhibit a dielectric dispersion. The graph on the right shows the normalized DEP and ROT responses calculated with the following parameters $\varepsilon_p = 10\varepsilon_0$, $\varepsilon_m = 78\varepsilon_0$, $\sigma_p = 0.3\,S/m$, $\sigma_m = 0.06\,S/m$ 19

Figure 3. The graph on the left shows the theoretical frequency dependence (complex permittivity, see eq. 17) of particles suspended in a medium that does not exhibit a dielectric dispersion. The graph on the right shows the normalized DEP and ROT responses calculated with the following parameters $\varepsilon_p = 240\varepsilon_0$, $\varepsilon_m = 78\varepsilon_0$, $\sigma_p = 0.02\,S/m$, $\sigma_m = 0.1\,S/m$ 19

Figure 4. Multi-shell model consist of n concentric spheres with different radius and complex dielectric constants 21

Figure 5. A suspension system with a characteristic dielectric constant ε^* encompasses a medium of ε^*_0 and particles of radius R_1 and dielectric constant ε^*_1 22

Figure 6. Formation of a pearl chain due to dipole-dipole interactions 24

Figure 7. The polynomial electrode configuration can be used to separate cells with the electrodes energized as shown in (a) or to apply a torque (and thus able to obtain the ROT spectrum from a particle) with the electrode setup shown in (b). 27

Figure 8. (a) Top view of the comb geometry used to produce travelling electric waves and (b) cross sectional view of the structure showing the direction of the traveling wave. A travelling wave electric field is induced by applying AC voltages of 90° separation. Particles that are less polarizable than the medium translate and rotate in a direction opposite to the travelling wave. 28

Figure 9. Interdigitated electrode configurations. (a) shows the comb geometry (also called interdigitated electrode structure), electrodes and gaps have the same dimensions. (b) shows the interdigitated-castellated electrode. Castells and fingers are equally separated. 29

Figure 10. Outline of the separation process of two type of cells using interdigitated electrodes. a) Positive DEP attracts one type of cells (black) to the edges of the electrode, while the other type of cells (white) are collected by negative DEP between the electrode gaps or on top of the electrode surface. b) A stream of flow flushes away the white cells because the negative DEP force is very weak. 30

Figure 11. Substrates double-coated with SJR7540 photoresist were characterized with an Alphastep profilometer. Measured value was 48.68 μm. 34

Figure 12. Schematic procedure for the creation of the master mold and for casting PDMS with PEEK posts serving as ports 36

Figure 13. DEP Microfluidic chip with ports showing the different parts composing it. 37

Figure 14. Schematic showing the adapter used to link the external fluidic network and the ports embedded in the chip (left). Picture showing the chip with one adapter connected to the inlet port (right). 37

Figure 15. Snapshots of (a) interdigitated electrode configuration, (b) interdigitated castellated electrode configuration, and (c) polynomial electrode configuration. 39

Figure 16. Alignment of polynomial electrodes and PDMS channels. Figure to the left shows the original design and the expected alignment. The width of the PDMS microfluidic channels is 20 microns. Figure to the right shows the built chip, PDMS layer containing microfluidic channels were aligned and bonded to the glass substrate with an accuracy of 20 microns. 41

Figure 17. Beads have adhered to the walls and bottom surface of the chamber as a result of hydrophobicity. Pictures were taken with a video camera (VK-C350, Hitachi, Tokyo, Japan) attached to an Olympus microscope (BH2-UMA, Olympus, NY, USA) 43

Figure 18. Bubble formation inside a microfluidic chip 43

Figure 19. Experimental setup for DEP devices. Details of the instruments are found in the text. 45

Figure 20. Electric field strength distribution in the x-y plane at z = 25 microns 10 Vpp for the quadrupole electrode structure, electrodes c) and d) are grounded. 48

Figure 21. Effects of the electric field on quadrupole structures at different frequencies on white sulfate beads. (a) shows the initial random distribution of white beads. (b) shows the response at the application of 5V at 1 kHz, the frequency was increased to 3.98 kHz in (c), 10 kHz in (d), 39.81 kHz in (e), 100 kHz in (f), 398.11 kHz in (g), 1 MHz in (h), and 3.98 MHz in (i). Beads experienced positive DEP forces up to frequencies of 3.98 kHz. Above frequencies of 10 kHz, beads experienced negative DEP forces. 49

Figure 22. Effects of the electric field on quadrupole structures at different frequencies on red CML latex beads. The chamber height is 48 microns. 51

Figure 23. Electric field strength distribution in the x-y plane at z = 25 microns 10 Vpp for the polynomial electrode structure. Electrodes are energized at a = 5V DC, b = -5V DC, c = 5V DC, d = -5V DC. 53

Figure 24. Microfluidic chip containing polynomial electrodes being used to study yeast cells. 54

Figure 25. (a) shows the random initial distribution of yeast cells at the polynomial electrode structure, beads are found specially on top of the surface of the right side electrode. At the application of 10 Vpp at 10 kHz yeast cells collected at the edges of the electrode caused by positive DEP forces and formed pearl chains as shown in (b) and pointed by the arrows. 55

Figure 26. 3D (above) and 2D (below) simulations of the electric field distribution in interdigitated electrodes. High electric field intensities are found at the edges of the electrodes. Weak electric field intensity is found between electrode gaps and on top of the electrode surfaces. 56

Figure 27. Separation of red (smaller) and white (larger) beads on interdigitated electrodes. (a) shows the initial random distribution of beads. At the application of 5V pp the red beads collected at the edges of the electrodes while the white beads were trapped on top of the electrode surfaces as shown in (b). Arrows on (c) point to red CML beads. 58

Figure 28. Electric field strength distribution at the castellated electrode (left) and zoom (right) in the x-y plane at z = 25 microns above the plane. Electric field intensity is stronger between electrodes while at the surface of the electrodes and at the electrode bays the electric field intensity is very weak. 59

Figure 29. Distribution of the electric field strength in the z-y plane (above); the height of the chamber is 50 microns. Three-dimensional electric field distribution (below) for the interdigitated castellated electrode structure 60

Figure 30. Effects of the electric field on castellated structures at different frequencies on white sulfate beads. Electrode gaps are 80 μm. White latex beads experienced positive DEP up to 10 kHz as shown in Figures a-c. At 20 kHz lonely beads collected on top of the electrodes and pearl chain formations collected at the electrode bays where the electric field is minima as shown in Figures d. Above 500 kHz all beads collected at the surface of the electrodes as a consequence of negative DEP........ 61

Figure 31. Effects of the electric field on castellated structures at different frequencies on red CML latex beads. Electrode gaps are 60 μm. Beads experiencing positive DEP were pulled very weakly at the electrode gaps upon the application of 10 Vpp at 5 kHz as shown in Figure a. At the application of 10 kHz, the positive DEP force decreased and the beads moved to the electrode bays. Above 20 kHz beads experienced negative DEP and collected on top of the electrodes as shown in (c) and (d). 62

Figure 32. Effects of the electric field on castellated structures at a fixed frequency on red CML latex beads. Electrode gaps are 60 μm. 63

Figure 33. Separation of red and white beads on castellated electrodes with gaps of 80 μm. Time elapsed is shown in seconds in the upper right corner on every frame. WLB indicates white latex beads. At time zero red beads are trapped at the electrode edges by positive DEP while white beads experiencing negative DEP are transiting through the electrode. The stream flow started to drag white beads along the electrodes until they were eluted (1 second to 10 seconds), because the drag force was enough to overcome the negative DEP force experienced by white beads. The last four frames show that only red beads remained trapped at the electrodes. 66

Figure 34. Diagram shows the main components of the custom function signal generator. The microcontroller can be programmed from a PC through the serial port. The digital potentiometer and the direct digital synthesizer are controlled via the I^2C protocol. I^2C signals generated in the microcontroller are multiplexed to the digital potentiometer and to the direct digital synthesizer. Dashed lines encompass the components in the custom PCB. 69

Figure 35. Schematic shows the connectors for the power supplies and to interface the microcontroller. I^2C signals are multiplexed to control the DDS and the digital pot. ... 70

Figure 36. Schematic comprising the frequency generator stage. The schematic shows the wiring of the DDS, Digital POT and the 50 MHz clock..... 71

Figure 37. Circuit showing the current-to-voltage op-amp configuration and the DC level shifter. See text for details. ... 72

Figure 38. Fixed gain amplification stage. Signal OUT1 is inverted using an inverter op-amp. Signals OUT1 and OUT_INV1 go then to unity gain buffers terminated on 75 ohm resistors. ... 73

Figure 39. Output signals from the custom signal function generator were measured with an oscilloscope. (a) shows the signal at 1 MHz 12 Vpp, (b) at 5 MHz 10.4 Vpp, (c) at 10 MHz 10.3 Vpp, and (d) at 20 MHz 10.8 Vpp. ... 74

Figure 40. Custom frequency generator setup showing the custom PCB and the microcontroller board. Modifications in the circuit are reflected in the schematics previously described, but not in the layouts. ... 74

Figure 41. Graphical User Interface used to display video, control the frequency generator, and store comments and parameters relevant to DEP experiments. See text for more details. ... 75

Figure 42. Diagram showing the creation of an image with close captioning. A static image containing information about the chip and the voltage and frequency applied to the chip is superimposed upon the video acquired with the video card (connected to a video camera). ... 77

Figure 43. (a) shows a schematic representation of a push-down valve. The flow layer is located underneath the control layer. (b) shows a schematic representation of a push-up valve. The flow layer is located atop the control layer. ... 80

Figure 44. Microfluidic chip design consisting of a microfluidic channel, a peristaltic micropump and two microvalves (not to scale). ... 81

Figure 45. (a) shows a picture of the patterned photoresist on the wafer before soft-bake, features are flat. The picture to the right, (b), shows the same patterned photoresist after it underwent a hard-bake for 15 minutes at 175°C, and shows that the features were rounded. ... 83

Figure 46. Schematic representation of the microfluidic chip fabricated by multilayer soft lithography. ... 84

Figure 47. Series of pictures showing the components of the microfluidic chip fabricated by multilayer soft lithography: a) micropump, b) microvalve, and c) a punched port. 85

Figure 48. Microfluidic chip fabricated by multilayer soft lithography showing the ports (stainless steel tubing), the location of the micropump, the microvalves and the microscope slide. 86

Figure 49. Experimental setup. A custom PCB (controlled through the parallel port of a PC), containing relays and solenoid valves, is connected to a pneumatic system (described below). The microfluidic chip is actuated with the pneumatic system. A camera and a microscope would be needed to observe the actuation of the microvalves. 86

Figure 50. Schematic showing the design of a pneumatic system used to supply regulated pressures to the microfluidic chip 87

Figure 51. Circuit schematic used to control the three- and two- way valves. (File Name: relays.sch) 89

Figure 52. Top view (left) and cross sectional view of the Cell DEP Capture Device showing microvalves, flow channels and electrodes, with dimensions. 92

Figure 53. Top view (left) and 3D view showing the control lines implemented in the control layer and the flow channels located in the fluidic layer. 93

Figure 54. A possible setup to drive DEP-based microfluidic chips integrated with microfluidic valves 94

Figure 54. Mask design for electrodes and PDMS channels. (File Name: fingers_castellated.tdb) 110

Figure 55. Top layer layout, frequency generator circuit. (File Name: one_freq_gen.brd) 112

Figure 56. Bottom layer layout, frequency generator circuit. (File Name: one_freq_gen.brd) 112

Figure 57. Bottom layer layout for the electronic valve control circuit. (File Name: relays.brd) 113

Figure 58. Top layer layout for the electronic valve control circuit. (File Name: relays.brd) 113

List of Notations

2D	Two dimensional
3D	Three-dimensional
$\vec{\nabla}$	Del Operator
ρ	Density
ε	Absolute Permittivity
ε_0	Permittivity of free space, $8.85 \times 10^{-12}\ \mathrm{CV^{-1}m^{-1}}$
ε^*	Complex Permittivity
σ	Conductivity
Φ	Volume Concentration
χ_e	Electric Susceptibility
v	Volume
ω	Frequency
$f(\varepsilon_p^*, \varepsilon_m^*)$	Clausius-Mossoti Factor
$\vec{s}$	Vector for the Dipole
$d\vec{\mathbf{E}}$	Differential of the Electric Field
$\vec{\mathbf{E}}$	Electric Field
$\vec{\mathbf{F}}$	Force
$\vec{\mathbf{p}}$	Dipole Moment
$\vec{\mathbf{P}}$	Dipole Moment per Unit Volume
$\vec{\mathbf{\Gamma}}$	Torque
$\vec{\mathbf{W}}$	Potential Energy
E	Dielectric Constant at High Frequencies

K	Conductivity at Low Frequencies
μFACS	Microfabricated Disposable Fluorescent Activated Cell Sorter
μl	Microleter
μm	Micrometer
μTAS	Micro Total Analysis Systems
AC	Alternating Current
BSA	Bovine Serum Albuminum
CE	Capillary Electrophoresis
CML	Carboxyl-Modified Latex
DC	Direct Current
DDS	Direct Digital Synthesis
DEP	Dielectrophoresis
DNA	Deoxyribonucleic Acid
EHD	Electro-hydrodynamic
EOF	Electroosmotic Flow
FACS	Fluorescent Activated Cell Sorters
FFF	Field Flow Fractionation
GND	Ground
GUI	Graphical User Interface
HDI	High Density Interface
HMDS	Hexamethyldisilazane
I^{2}C	Inter IC Serial Bus
ID	Inner Diameter
IPA	Isopropanol

J	Joule
LC	Liquid Chromatography
MACS	Magnetic-Activated Cell Sorters
MEMS	Microelectromechanical Systems
N_2	Nitrogen (gas)
NC	Normally Closed
NO	Normally Open
O_2	Oxygen (gas)
OD	Outer Diameter
Pa	Pascal
PCB	Printed Circuit Board
PCR	Polymerase Chain Reaction
PDMS	Polydimethyl Siloxane
PMMA	Polymethyl Methacrylate
POT	Potentiometer
Psi	Pound per Square Inch
ROT	Angular Velocity
TAS	Total Analysis Systems
TWD	Travelling Wave Dielectrophoresis
UV	Ultraviolet
W	Watts

Chapter 1

1 INTRODUCTION

The combination of chemical analyses, biological assays, and medical diagnostics are indispensable in developing new drugs, for therapeutic purposes, as a screening tool for disease detection, or simply as a way to gain a better understanding of biology and thus our world. On the other hand, the microelectronics and semiconductor industry provide tools and well-known techniques which make it possible to work with and manipulate organisms and particles at the micron scale and smaller. Integrating those areas have given birth to "labs-on-a-chip" research which attempts to integrate diagnostic tools and laboratory instrumentation in miniaturized systems, making them amenable to mass production, reducing sample consumption, allowing the possibility of on-site operation, and facilitating their operation.

The detection, analysis, and sorting of cells is of vital importance for the early detection, prevention, quantification, and diagnosis of abnormalities in cells which can lead to cancer or other diseases. Cell sorting is a major component towards the integration of a "lab-on-a-chip" and is the first step in a sample-preparation for molecular diagnostics [1]. The immediate treatment of the cell within the lab-on-a-chip after separation or isolation would reduce sample loss and contamination. Furthermore, it could have a huge impact on the time, automation and accuracy of the test.

This thesis describes the development of a bio-particle separation and capture microdevice. The device use dielectrophoretic forces to separate cells or beads and microvalves to aid in the control and regulation of the fluids going through the channels.

This chapter commences with a brief description of the different mechanisms currently employed to separate bioparticles. We then describe the relevance and impact of micro-total analyses systems (μTAS) or lab-on-a-chips in our everyday life as well as the importance of integrating cell separation methods within them. Next, we review the materials commonly used to fabricate lab-on-a-chips as well as the most important components that are helpful in the separation of bioparticles. We conclude with an overview and general description of the following chapters.

1.1 Cell Separation

There are several techniques to separate cells. Mechanical, electrical, electrochemical, pneumatically actuated or optical methods can be used in conjunction or in stages to sort them. Some of these techniques are more appropriate to purify and filter out a population of target cells, others give better resolution, others are optimal for single cell detection, and some of them are faster but expensive. The selection of the mechanism to separate cells will depend on the needs of the application. A subsequent step after a cell has been separated and isolated would be to perform a cell lysis or cell disruption to release the nucleic acid [2]. Methods for lysing are discussed in [1].

In this section, we explore common techniques to separate cells implemented in microdevices and discuss them when appropriate. Centrifugation, electrophoresis and fluorescence- and magnetic-activated-cell sorting are the most common techniques used for cell separation. However, new improvements in separation resolution, cell purity, sample size, device cost, and portability within these techniques are difficult to achieve[3].

1.1.1 Electrophoresis

It is well known that cells, as well as DNA, typically have a net electrical charge. By suspending them in a liquid reagent and applying an electric field, cells or DNA will move at different velocities depending on their net charge; variations in their mobility lead to their separation. This phenomenon is given the name electrophoresis[4]. Section 1.4.1.2 discusses this in more detail.

1.1.2 Flow-Field Flow Fractionation

Field Flow Fractionation (FFF) works by combining a stream of liquid in a microfluidic channel (under laminar flow conditions) and a field or gradient (thermal, electrical, flow, sedimentation, and lastly dielectrophoresis) acting perpendicular to the flow [5-8]. The properties and composition of the cells or colloids respond differently to temperature gradients, electric fields, viscous forces or acceleration forces. These forces interact with the solutes and serve to drive the particles into specific stream laminae in the microfluidic channel. The separation is caused by the unequal velocities of the stream laminae.

Flow FFF is used to separate particles based primarily on their size (diameter of the particle)[9]. Sedimentation FFF separates particles based mainly on their effective mass, particle volume, diameter, and differences in density between particles and the liquid[10]. Thermal FFF can discriminate polymers by differences in their molecular weight[7]. Net charges or electrophoretic mobility are variables used to separate particles in Electric FFF[10]. Dielectrophoretic forces can work in conjunction with FFF to discriminate cells based on their dielectric fingerprint[11].

1.1.3 Filtration

Filtration is a technique to separate cells based on their size either in a large or small scale. Physical barriers (porous membranes or hollow fibers) are used to retain virus, bacteria, viable and nonviable cells, or large debris[12]. Porous membranes (pore sizes ranging from 0.1 to 5 μm [13]), for example, retain particles bigger than their pore sizes. However, fouling and clogging problems are often reported as the major drawback of this technique; and as for the separation of cells, much of the success of this technique is dependent on the deformability of cells, cell concentration, viscosity of the medium, size of the filter, and the pressure needed to make them pass through the barrier [14].

Filtration has been transferred to the microscale realm by spacing an array of posts or a series of channels at specific gaps inside a microchamber. This technique has been used to separate white blood cells from red blood cells. Weir-type filters and comb shaped filters fabricated in glass and silicon are other options for filtration [1].

1.1.4 Fluorescence Activated Cell Sorters

Biological cells can be separated based upon the light scattering produced by the color of cells previously labeled with a fluorescent dye and by their size. Cells can be separated one by one, if desired, in to specific containers[15]. However, these Fluorescence Activated Cell Sorters (FACS) are bulky, expensive, and are operated by a trained technician.

A microfabricated disposable Fluorescent Activated Cell Sorters (μFACS) made in Polydimethyl Siloxane (PDMS) has been demonstrated by Quake *et al.* [16] to sort, recover, interrogate and trap cells. It consists of three channels joining at a T junction and three ports; one inlet for the input sample and two outlets for the waste and collection. Electrokinetic flow moves the fluorescently labeled cells (*Escherichia coli* cells) and an optical system (a photomultiplier and a laser beam) is used to detect the fluorescent emission of the cells at the detection point. Upon detection, voltages are switched to divert the detected cell to the collection well. This design was later modified to include multilayer soft lithography based monolithic valves. Electrokinetic-driven flow was replaced by pressure-driven flow which resulted in an improvement of the sorting accuracy, surface chemistry, buffer compatibility, automation and is less harmful to the cells [17]. With this sorting mechanism it is possible to separate 100, 000 cells per hour and it is believed that this FACS will be able to yield close to 100% recovery with further modifications.

1.1.5 Magnetic Activated Cell Sorter

Another approach to separate cells is using magnetic fields. A microfluidic chip containing an array of nickel posts magnetized by an external neodymium-iron-boron magnets was built by the Whiteside's group at Harvard University. Using this method, two types of beads, magnetic beads and nonmagnetic beads, diluted in a diamagnetic solution were separated. Magnetic beads were retained at the energized posts and dyed polybeads continued to flow until they were eluted. Each post could capture up to 50 beads. The magnetic microfiltration system could capture more than 95% of the magnetic beads because some beads remained attached to the posts after the removal of the external magnetic field [18]. Magnetically-Activated Cell Sorters (MACS) follow the same principle, instead of posts, a separation column or a permanent magnet placed next to the channel can retain cells previously labeled with antibody-labelled superparamagnetic beads [19].

1.1.6 Dielectrophoresis

Dielectrophoresis (DEP) has been demonstrated to be an effective, reliable and inexpensive method of separating human cancer cells from healthy cells [3, 20, 21], to trap cortical rat neurons [22], to organize particles in two and three dimensions [23], and to sort any kind of particles, plant, animal and human cells, as well as bacteria and viruses [24-26]. Recently, a group from three different universities demonstrated the concentration and patterning of single- and double-stranded DNA using dielectrophoretic methods [27].

DEP accounts for the motion of polarizable particles in the presence of inhomogeneous electric fields. All types of cells exhibit exclusive physiological and morphological states which give them particular dielectric fingerprints [2]. DEP can provide information about the organelles composing a cell: surface conductance, membrane capacitance and conductance, cytoplasmic conductance. Furthermore, DEP can determine differences in the cell's physical state: age, cycle phase, and state of health[28].

These dielectric properties vary in accordance to the magnitude and frequency of the applied electric voltage, the dielectric characteristics of the suspending medium and the geometry of the electrodes. Under the appropriate set of conditions, any bio-particle will react in one of two opposite ways, either it will translate to strong electric fields or it will move to weak electric fields. The successful separation and detection of cells differing only in their cytoplasm composition (size, molecules, proteins) indicates the versatility and sensitivity of dielectrophoresis[29].

DEP is not limited to differentiating based on cell size because different type of cells even with the same radii exhibit distinct dielectric characteristics. Cells do not require any labeling [30] and the same simple electrode design can be employed to sort different types of cells [31]. DEP has shown the potential for single-cell studies and cell characterization [32] but also cells can be studied statistically; it is non-invasive; it can monitor the location and quantify particles within a microdevice [33]. Electrodes can be individually addressed so different cells can be separated within reduced spaces.

Dielectrophoresis has found many applications: to separate particles, to characterize and determine the structure of bioparticles by electrorotation and levitation, to trap and retain particles in specific areas, and to align cells before their electrofusion[34]. DEP can also aid in the study of cellular changes due to induced cell differentiation, miotic activation, toxin exposure, and cell death [20]. Furthermore, DEP can help in the understanding of the physico-chemical properties of colloidal particles and in the screening of microorganisms in water [35].

The main drawback of DEP is that buffers with specific conductivities are required, which means that the original sample has to be pre-treated or diluted with appropriate chemicals to obtain the desired conductivity. Other phenomena can affect the separation, sensitivity, efficiency, and throughput of DEP devices such as Joule heating, AC electro-hydrodynamic flow and cell-cell interactions [22].

Joule heating due to high electric fields and high conductivity buffers can be harmful or deadly for cells and can affect the outcome of the experiments because of the high temperatures produced. Steady fluid flow is created by the interaction of non uniform ac electric fields (DEP forces) with the induced charges in the electric double layer across the surface of coplanar microelectrodes structures. This phenomenon is known as AC electro-hydrodynamic (EHD) flow and occasions the fluid to move from high strength field regions (located at the edges of the electrodes) onto the surface of the electrodes [36, 37]. This issue would need to be addressed if DEP is to become a widespread and useful technique [1].

Nonetheless, advantages offered by DEP outweigh its limitations. Furthermore, it is suitable to microfabrication and to integration with micro biological assays and offers the possibility to automate complex experiments [33]. Because of the aforementioned reasons, we chose dielectrophoresis as the sorting mechanism to separate cells.

1.2 Micro-Total Analyses Systems

The prompt and accurate detection of chemical or biological parameters is sometimes crucial for biological analysis and medical diagnostics (cell, protein, and DNA analysis), pharmaceutical applications (drug discovery), environmental settings (e.g. portable water screening equipment), chemicals detection (toxic substances), and many other fields. Chemical and biological analysis methodologies are generally comprised of the following sequential steps: sampling, sample transport, sample pre-treatment, separation and detection of the analyte. These total analyses are carried out using conventional and bulky laboratory instrumentation in a reliable fashion, but without the possibility of in-situ analyses[38].

Reducing the size of the total analysis systems (TAS) offers many advantages like mass production, portability and on-site operation, ease of use, low sample consumption and high stability. It is also expected that miniaturization could automate most of the operations and thus eliminate manual labour, reduce processing and analysis-time, diminish pipetting errors that occur in the intermediate stages of the assay, and improve the reproducibility of the results [39].

Present technologies such as microlithography, micromachining, micro - electromechanical systems (MEMS), and most recently, nanotechnology, have allowed and aided in the fabrication, implementation, and integration of biological analysis systems at the micron-scale. This would allow components of a modern laboratory like fluid handling, pumps, separators, valves, electrophoresis columns, reactors, heaters, and sensors to be readily integrated onto one single device [40]. Micro Total Analyses Systems (μTAS), lab-on-a-chip, microfabricated bioanalytical devices or microfluidic systems are terms often employed to name these micro devices capable of performing multiple biological assays.

Microfluidic devices demand the presence and integration of mechanical, electrical, electrochemical or optical elements as detection and control mechanisms, and in general to increase the throughput of the miniaturized total analysis systems. Lab-on-a-chip functions would also include sample preparation, handling, mixing, incubation, separation and concentration, quantification, sorting, and transportation. Trying to integrate different assays within the same microfluidic chip, and to find a balance and a perfect fit among them is a challenge as issues such as material compatibility, fabrication feasibility, buffer compatibility (pH and ionic strength), conductivity, and cost have to be considered when designing a device. At least four things have to be kept in mind when considering the design of a micro analysis system: it is has to be sensitive, inexpensive, robust and easy to operate.

1.3 Materials for Micro-Total Analyses Systems Technologies

μTAS devices need to be made out of materials that do not lead to any hazard to or contamination of the biological assays being performed within them. Currently, most lab-on-a-chips are a combination of different materials (such as the ones described below). μTAS research efforts are aimed to integrate the different biological tools, procedures, and assays that are commonly found in the macroscopic world within only one material. Such a material has to be inexpensive, needs to be amenable to microfabrication, and has to offer the possibility to embrace as many components as possible.

1.3.1 Silicon and Glass

Silicon, glass and some polymers are commonly used to fabricate microfluidic devices. Silicon is abundant, widely characterized and employed in micromechanical structures, but it is not optically transparent. Glass offers the advantage of being optical transparent so assays can be monitored through a microscope if required. However, microfabrication processes developed for glass and silicon are time consuming, expensive, and require a clean-room environment. Furthermore, its higher cost can increase the price of a chip dramatically [41]. It is thus apparent that it is crucial to have materials that are inexpensive so that the devices can be disposed of, preventing any cross contamination from other samples.

1.3.2 Polymers

In contrast, polymers (Polymethyl methacrylate-PMMA, polycarbonate, polystyrene, polyester copolymer-PETG, among others) and soft polymers (Polydimethyl

Siloxane-PDMS) offer the possibility of fabricating disposable and single-use devices; manufacturing processes are less expensive because channels can be formed by hot embossing or molding rather than etching [41]. The production of nanometer features is also possible [42]. Most polymers cannot withstand high temperatures and they are often incompatible with organic solvents and low molecular weight organic solutes [41].

1.3.3 PDMS

PDMS is 50 times cheaper than silicon on a volume basis [42]. PDMS permits rapid prototyping. It also can be sealed permanently or reversibly to different materials by surface oxidation (plasma and corona discharge) [43] or by heating it for extended periods of time [44].

PDMS is gas permeable, water impermeable (in discussion [45]), and it is non-toxic to proteins and cells. It is thermally stable up to 95°C (required for biological assays), it cures at low temperatures; and it is possible to control its surface chemistry. Its Young's modulus is five orders of magnitude smaller than hard materials (silicon and silicon nitride) [46]. Finally, it is optical transparent in the UV/visible spectra down to 280 nm [47, 48].

Micro-biological assays have been already realized in PDMS. This list is abundant and includes: immunoassays, capillary electrophoresis, enzymatic assays, sorting and manipulation of cells and DNA, polymerase chain reaction, capillary electrophoresis and 2D gel-electrophoresis, liquid chromatography, patterning of biological and nonbiological materials on PDMS substrates, sensors, and patterning of proteins and cells [19, 47]. Moreover, a list of different components has been envisaged in PDMS such as cell-based biosensors, analyte detectors, switches, mixers, embedded organic membranes and gels, magnetic filters, and microreactors [19, 43, 47, 48]. PDMS-based systems are currently an area of intense research and rapid progress. A future integration of all of these components into one single chip is in fact possible but some concerns such as buffer compatibility (pH, conductivity, and the concentration of active particles, osmoles, in the solution) at different stages and running conditions still need to be resolved.

Table 1. Physical and Chemical Properties of PDMS [43-45, 49-51]

Young's Modulus	~750 kPa
Density	920 kg/m^3
Shear Modulus (high elasticity)	up to 3 MPa
Optical UV detection (good visibility)	240 to 1100 nm
Surface tension of wetting	24 mN/m
Water contact angle (virgin – hydrophobic)	> 100°
Glass transition temperature (low)	-125 °C
Thermal conductivity (low)	0.17 W/ mK
Diffusion coefficient of water at 25 °C	~ 2 x 10^{-9} m^2/sec
Water uptake capacity (high for polymers)	0.38 % (w/w)
Activation energy for water diffusion	14 kJ/mol
Dielectric strength (high)	~14 V/µm

We chose PDMS as our fabrication material for these reasons. In addition, our lab is exploring the implementation of polymerase chain reaction (PCR) and other assays in PDMS.

1.4 Core Components of Micro Total Analysis Systems

1.4.1 Microfluidic Pumping

Fluid transport in microfluidic channels is generally created using syringe pumps, peristaltic pumps, or electrokinetic flow. Other methods have been proposed in the last ten years such as: AC electrokinetics, electrochemical bubble generation, centripetal pumping, acoustic pumping, or magnetic-hydrodynamic pumping systems. Due to their actuation mechanisms, centripetal-, pressure- and electrokinetic-driven flow can be implemented either in soft or hard materials. The rest of the pumping methods have to be realized in glass or silicon, or require a piezoelectric actuator [1, 52]. The most popular methods are briefly discussed in this section.

1.4.1.1 Macro- and Micro- Pumps

Pressure forces generated by mechanical pumps (motor-driven syringes or peristaltic pumps) are traditionally used to handle liquids in the channels because of the simplicity of implementation. Applying pressure or vacuum to the ports inside a microfluidic channel can create a laminar flow if the Reynolds numbers are less than 2300. This laminar pressure-driven flow has a parabolic profile where the velocity increases from zero near the wall to a maximum towards the center of the flow. The flow profile can be characterized and determined beforehand if the dimensions of the channel, velocity, density and viscosity are known [53].

Micromechanical pumps can pump fluids in the order of 1 $\mu L\ min^{-1}$ while exhibiting low dead volumes and fast response but at the cost of complex fabrication processes and significant space and power consumption [1]. One of the advantages of macro- and micro- mechanical pumps is that they work with different solvents and materials even those that are not electrically conductive [19]. However, pressure driven flow is less effective as the channel dimensions diminish. Small scales require extreme pressures which makes these pumps more difficult to integrate with microfluidic chips.

1.4.1.2 Electrokinetic Pumps

Electrokinetic pumping remains one of the most popular methods to control and move liquids in a network of microchannels and it is commonly used in capillary electrophoresis (CE). It is easy to implement: two electrodes (wires) are immersed at separated reservoirs to create flow, and fluid is controlled by switching the voltages on and off. Fluid flow is caused by electroosmotic flow (EOF).

Electrophoresis separates molecules, ions, or cells based on their size, shape, and electric charge. At the application of a voltage difference, charged molecules or ions migrate through a solution to electrodes having different polarities than the molecule. Mobility is determined by their charge (the electric force is proportional to their charge and thus highly charged particles would travel faster through the medium) and by their size and shape (small molecules have less frictional drag and thus travel also faster than heavier molecules).

Electroosmosis is the result of the interaction of surface properties in capillaries and the solution. Glass capillary walls are negatively charged because of deprotonation silanol groups present and because of the absorption of ions from the solution (which is dependent on the pH of the medium). As a result, there is an attraction of cations from the medium to the surface until a double layer of ions is formed that maintains the charge balance. This accumulation of ions causes a potential difference (zeta potential) through a thin layer. The application of an electric field across the capillary drives the double layer of ions towards the cathode. The ions then would drag the bulk solution by viscous forces creating uniform plug-like fluid flow [54]. Electroosmotic flow results in flat velocity profiles as compared to laminar flow.

Electrokinetic pumping is efficient and relieves the need for mechanical valves and pumps when the microchannel architecture is simpler; so it is not surprising that some fluidic methods have been adapted to exploit this technique [55, 56]. However, it has drawbacks such as buffer incompatibility, voltage settings modifications (due to ion depletion), diffusion, electrolytic gas bubble formation, solvent evaporation, and non-uniform flow for the species due to different electrophoretic mobilities [19]. An increase in the complexity of the microchannel architectures and parallel processing of analytes would demand more accurate control than is possible to achieve with electrokinetic pumping because the number of electrodes required would increase as more control is needed; it would be simpler to incorporate a microfluidic valving system [56].

1.4.1.3 Induction Electrohydrodynamic Pumps

Alternating electric fields can also pump fluids in a micro-channel. Induction Electro-hydrodynamic (EHD) pumps generate induced charges at the interface formed by an electrode surface and a dielectric fluid. Energizing a planar electrode array with voltages with the same magnitude but out of phase can produce a travelling electric field; this travelling wave would drag or pull the induced charges along the wave direction [54]. This technique requires precise control of the permittivity and the conductivity of the medium. Furthermore, it is limited to fluids of low conductivity and precautions have to be taken to avoid any non desired heating [55]. Another drawback is their use of high operation voltages [57] that are deadly for cells.

1.4.2 Microfluidic Valves

Microvalves can be categorized as passive and active valves. Active valves contain an actuator: piezoelectric, cantilever, coil spring or thermopneumatic; passive valves are those without an actuator [1, 55, 56]. Microvalves are needed in different applications such as in Liquid Chromatography (LC) on a chip (permitting faster analyses and use of small samples), in large-scale integrated systems (integrating a massive amount of them), and on-chip actuated systems (incorporating valves and other components inside the chip) [58]. This section will briefly mention some of the implemented microvalves.

1.4.2.1 Freeze-thaw Valves

A plug of a liquid flowing in a channel can be frozen, thus stopping the liquid's normal course. Thawing the plug resumes its normal flow. Freeze-thaw valves exploit this fact. Their advantages are their zero dead volume and the absence of moving parts. The mainly hindrance is their response time (20-45 seconds)[55].

1.4.2.2 Micromechanical valves

Micromechanical valves have also been implemented [52, 55].. Most of them are fabricated in silicon and use actuators such as piezoelectric plates or cantilevers. They share the same drawbacks as the micromechanical pumps discussed in a previous section. Silicon, glass, and metals were the common choice in the infancy of microfluidics, but the fabrication process and the assembling of different parts was rather complex and involved many steps which in turn led to longer times to fabricate a single microfluidic device, involved more equipment, produced greater number of errors, increased the cost, and required dedicated facilities such as a clean room [58]. However, glass is still commonly used because of its optical properties.

Micro-electromechanical approaches are not the optimum approach for devising biological assays for the reasons explained above. The trend is to use plastics, elastomers, and hydrogels, because they allow for quick development, rapid prototyping, less equipment, and low costs[59].

1.4.2.3 Hydrogel and Surface Tension Valves

Chemically sensitive hydrogels can undergo volume changes depending on the conditions of the surrounding medium without external power sources[53]. This mechanism has been exploited by Beebe *et al.* to create pH sensitive on-chip valves. Variations in the pH causes these 'smart gels' to change from neutral to charged because of the protonation of amine groups or the deprotonation of acid groups, and thus experience a volume transition [60]. They integrated pH responsive valves in the form of posts to control the flow in a microfluidic channel. These valves expanded or contracted due to osmotic pressure exerted by mobile ions. The main disadvantages are that they cannot handle high pressures, they can absorb part of the sample, and have a slow response [19, 58, 61, 62]. The fastest time response for a 2-D hydrogel microvalve was 12 seconds [60].

Surface tension gradients in the form of hydrophobic and hydrophilic patches can be also used as valves. Hydrophobic surfaces can be used to preclude the passage of liquid unless the pressure driving it is enough to overcome the surface tension, and thus open the valve. Closing the valve requires a complete drying out of the channel, a significant disadvantage [58].

1.4.2.4 Multilayer Soft Lithography Microvalves and Micropumps

Multilayer soft lithography is based on soft lithography and offers the advantage of bonding multiple patterned layers of PDMS by changing the ratio of the concentration of the components that make the elastomer. The group that developed this technique has demonstrated its use to create microvalves and micropumps [44].

Monolithic microfabricated microvalves and micropumps by multilayer soft lithography are easy to fabricate, and permit rapid prototyping. They are also inexpensive, exhibit fast responses, show ample fluid compatibility, prevent the formation of air bubbles [19], and are scalable and leak-proof [63]. Some disadvantages of these microvalves are that they cannot handle high pressures and cannot interact with strong solvents[58]. Nonetheless, massive integration and the ease with which they can be implemented are their main advantages when compared to the aforementioned mechanisms. For these reasons, and because our system requires a simple flow-control mechanism, we decided to work with multilayer soft lithography.

1.5 DEP Devices Implemented on PDMS

To our knowledge microfluidic dielectrophoretic devices have not yet been fully integrated with PDMS. Up until now, the fabrication of dielectrophoretic devices consisted of several components made of different materials, and this hinders a possible integration with assays already implemented in PDMS as discussed in section 1.2.1.

A typical DEP microdevice consists of gold or platinum electrodes patterned on glass, a spacer bonded or glued to the glass, to be finally covered by a microscope slide. Spacers are made out commonly of SU8 photoresist [24, 64], polyacetate [65], HybriWell cover (manufactured by Grace Bio-Labs Inc., OR, USA)[66], PMMA [67], silicon rubber[68], Teflon [69] or O-rings[31]. In the best cases, complete fluidic channels were made in glass, but this approach is expensive and time-consuming. Two holes drilled in the microscope slide or the glass substrate serve as inlet and outlet ports[70, 71].

This configuration complicates the collection of sample because non-desired cells could be eluted first and some of them could then get trapped at the outlet or remain in the channels and get mixed with the rest of the target cells, which in turn could give rise to erroneous results. There is also the interaction of the different materials composing the microfluidic chip: glass, spacer, and the microscope slide. The only advantage we found to these setups was that they are easier to clean and can be mounted and demounted several times, so different channel configurations patterned in the spacer can be tried out using the same patterned glass substrate. Nonetheless, this hinders future integration with other assays that have been already fully integrated with other materials, especially PDMS; also integration of components such as pumps or valves would be difficult to integrate for the same reasons. Today, PDMS has been used very little in DEP applications.

1.6 Scope of the Thesis

It is the scope of this thesis to explore the viability of the integration of dielectrophoresis with micro technologies to separate cells and to isolate a single cell and deliver it to a well or reservoir thereby producing a functional lab-on-a-chip. For this purpose we have selected PDMS as the primary material in our fabrication process to create the microfluidic channels, and fabricate microvalves and micropumps in order to provide control over the microfluidic channels. PDMS is very promising and researchers are finding new applications and creating new components out of it every day.

The thesis is divided in eight chapters. Concepts and a general discussion of the thesis were explained in this chapter. The next chapter gives an overview of the dielectrophoretic theory as well as some of the electrical characteristics of the cell.

The following chapter explores the way in which different electrode geometries are used to detect and manipulate single cells or to sort them, as well as how some of them have been combined with other different techniques to enhance sensitivity. In the fourth chapter, the fabrication processes with some of the issues and concerns encountered are explained. An experimental chapter follows, giving the results obtained. The chapter entitled Hardware and Software Tools presents the design and assembly of a custom function generator and a description of the graphical user interface (GUI) used to capture video and record data from the experiments. The next chapter describes the design and creation of micropumps and peristaltic pumps fabricated by multilayer soft lithography. Finally, the conclusion explores possibilities for future work.

Chapter 2

2 THEORETICAL BACKGROUND

2.1 Dielectrophoresis

This chapter explains what dielectrophoresis is. We start the first section by calculating the electric forces and torque experienced by a homogeneous particle of a specific permittivity subjected to an inhomogeneous electrostatic field [72, 73]. The next section shows how to obtain the force and torque experienced by a homogeneous particle with a complex permittivity subjected to an alternating electric field. Following these sections, we study the effects of alternating fields on cells modeled as a series of concentric spheres; this section is key for the understanding of dielectrophoresis.

2.1.1 Homogeneous Sphere in an Non Uniform Electrostatic Field

We start this chapter by giving a pedagogic example of what happens to a sphere placed in a non-homogeneous electric field so as to grasp the basic idea of dielectrophoresis. We explore the dipole moment produced by the bound charges of a homogeneous sphere of radius R once it has become polarized assuming that there is no external electric field. The single dipole moment is given by:

$$\vec{\mathbf{p}} = \frac{4}{3}\pi R^3 \vec{\mathbf{P}} \tag{1}$$

where $\vec{\mathbf{P}}$ is the dipole moment per unit volume and the electric field $\vec{\mathbf{E}}$ produced by the sphere can be written as

$$\vec{\mathbf{E}} = -\frac{1}{3\varepsilon_0}\vec{\mathbf{P}} \tag{2}$$

where ε_0 is the permittivity of vacuum. The dipole moment is thus proportional to the volume of the sphere $v = \frac{4}{3}\pi R^3$. However, to enable the polarization, an electric field would have had to be applied, which would induce the bound charges. Thus, the polarization would be, for a homogeneous sphere, proportional to the electric field, and can be written as:

$$\vec{\mathbf{P}} = \varepsilon_0 \chi_e \vec{\mathbf{E}} = (\varepsilon - \varepsilon_0)\vec{\mathbf{E}} \tag{3}$$

$$\vec{\mathbf{p}} = \frac{4}{3}\pi R^3\left(\varepsilon - \varepsilon_0\right)\vec{\mathbf{E}} = v\left(\varepsilon - \varepsilon_0\right)\vec{\mathbf{E}} \qquad (4)$$

where χ_e is the electric susceptibility of the material, ε_0 is the permittivity of vacuum, ε is the absolute permittivity of the sphere, and $\varepsilon - \varepsilon_0$ is the polarisability.

Now, if we bring this sphere as it is (polarized) into the presence of a uniform electric field, forces on both ends would be equal in magnitude but opposite in direction and as a consequence would cancel each other. It will be impossible for the sphere (dipole) to displace to other places. However, the dipole would try to align (and thus rotate) parallel to the electric field since forces at its ends have different directions. We can quantify the torque $\vec{\mathbf{\Gamma}}$ induced in the dipole:

$$\vec{\mathbf{\Gamma}} = \vec{\mathbf{p}} \times \vec{\mathbf{E}} \qquad (5)$$

In the case where a nonuniform electric field is present, there would be a net force on the dipole, since forces on both ends would differ, then

$$\vec{\mathbf{F}} = q\left(\vec{\mathbf{E}}_+ - \vec{\mathbf{E}}_-\right) = q\left(d\vec{\mathbf{E}}\right) \qquad (6)$$

where $d\vec{\mathbf{E}}$ is the differential of $\vec{\mathbf{E}}$ and $\vec{s}$ is the vector for the dipole, thus

$$d\vec{\mathbf{E}} = \left(\vec{s} \cdot \vec{\nabla}\right)\vec{\mathbf{E}} \qquad (7)$$

as a result,

$$\vec{\mathbf{F}} = \left(\vec{\mathbf{p}} \cdot \vec{\nabla}\right)\vec{\mathbf{E}} \qquad (8)$$

But we know from equation (4) that $\vec{\mathbf{p}}$ is proportional to $\vec{\mathbf{E}}$, then substituting equation (4) into equation (8) we obtain that the force is given by:

$$\vec{\mathbf{F}} = v\left(\varepsilon - \varepsilon_0\right)\left(\vec{\mathbf{E}} \cdot \vec{\nabla}\right)\vec{\mathbf{E}} = \frac{1}{2}v\left(\varepsilon - \varepsilon_0\right)\vec{\nabla}\left|\vec{\mathbf{E}}\right|^2 \qquad (9)$$

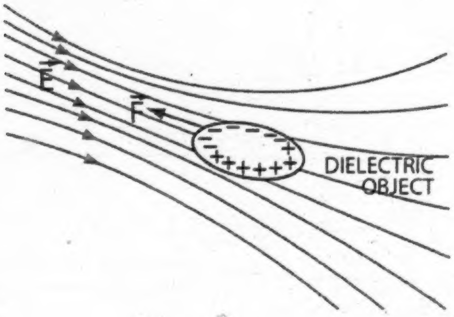


Figure 1. A dielectric object subject to the presence of an inhomogeneous electric field experiences a net force to regions of high strength electric field.

There is a net force because the field is changing from point to point; as a consequence, the sphere will be drawn from regions where the field is weak to regions where it is stronger. Therefore, the force exerted on a sphere would be proportional to the dielectric constant of the sphere, to its volume and to the gradient of the square of the applied electric field. The equation stated above for the calculation of the torque is still valid for nonuniform electric fields.

The formulas derived above consider only vacuum, but if we were to replace vacuum by a liquid with medium permittivity ε_2, the dipole moment experienced by a sphere with permittivity ε_1, equation 4 can be restated as [74]:

$$\vec{\mathbf{p}} = v\varepsilon_1\left(\frac{\varepsilon_2-\varepsilon_1}{\varepsilon_2+2\varepsilon_1}\right)\vec{\mathbf{E}} \tag{10}$$

and consequently the force is given now by:

$$\vec{\mathbf{F}} = \frac{3}{2}v\varepsilon_1\left(\frac{\varepsilon_2-\varepsilon_1}{\varepsilon_2+2\varepsilon_1}\right)\vec{\nabla}\left|\vec{\mathbf{E}}\right|^2 \tag{11}$$

Dielectrophoresis is fully described by this formula; the next section presents a more complete derivation of dielectrophoresis when complex permittivities and alternating electric fields come into play.

2.1.2 Homogeneous Sphere in an Alternating Non Uniform Electric Field.

It has been well established that the force $\vec{\mathbf{F}}(\omega)$ and torque $\vec{\Gamma}(\omega)$ experienced by a homogeneous sphere of radius R subjected to an alternating electric field $\vec{\mathbf{E}}$ of angular frequency ω in a lossy medium is given by [75]:

$$\vec{\mathbf{F}}(\omega) = \mathrm{Re}\left(\vec{\mathbf{m}}(\omega)\right)\frac{\nabla\vec{\mathbf{E}}_{RMS}^{2}}{2\vec{\mathbf{E}}} \tag{12}$$

Specifically, for an electric field in the z plane, $E(x,y,z)\cos(\omega t)\vec{\mathbf{a}}_z$, equation (12) can be restated as:

$$\vec{\mathbf{m}} = 4\pi\varepsilon_m\left[\mathrm{Re}\,f(\varepsilon_p^*,\varepsilon_m^*)\cos(\omega t) - \mathrm{Im}\,f(\varepsilon_p^*,\varepsilon_m^*)\sin(\omega t)\right]E(x,y,z)\vec{\mathbf{a}}_z \tag{13}$$

whereas the torque experienced by the particle is derived as:

$$\vec{\Gamma}(\omega) = -\mathrm{Im}\left(\vec{\mathbf{m}}(\omega)\right)\vec{\mathbf{E}} \tag{14}$$

where $\vec{\mathbf{m}}(\omega)$ is the dipole moment experienced by the particles and is given by:

$$\vec{\mathbf{m}}(\omega) = 4\pi\varepsilon_m f(\varepsilon_p^*,\varepsilon_m^*)R^3\vec{\mathbf{E}} \tag{15}$$

where $f(\varepsilon_p^*,\varepsilon_m^*)$ is known as the Clausius-Mossoti factor and is defined as:

$$f(\varepsilon_p^*,\varepsilon_m^*) = \frac{\varepsilon_p^* - \varepsilon_m^*}{\varepsilon_p^* + 2\varepsilon_m^*} \tag{16}$$

where ε_p^* is the complex permittivity of an homogeneous sphere and ε_m^* is the complex permittivity of the medium.

The complex permittivity ε^* [74] is obtained experimentally and is defined as a function of the frequency ω:

$$\varepsilon^* = \varepsilon - j\left(\frac{\sigma}{\omega}\right) \tag{17}$$

where ε is the in-phase (or real) permittivity (dielectric constant) and σ is the dielectric conductivity (σ/ω is known as the out-of-phase component) of the material due to the migration of charge carriers or to any other energy-consuming process.

The time-averaged potential energy of the particle is:

$$\vec{\mathbf{W}}(\omega) = -\vec{\mathbf{m}}(\omega) \bullet \vec{\mathbf{E}} \qquad (18)$$

so that the average potential energy of the polarized particle for low loss materials (i.e. $\varepsilon \gg \sigma/\omega$), is

$$\left\langle \vec{\mathbf{W}} \right\rangle = -2\pi\varepsilon_m r^3 \operatorname{Re}\left[f(\varepsilon_p^*, \varepsilon_m^*) \right] \vec{\mathbf{E}}_{RMS}^2 \qquad (19)$$

thus,

$$\vec{\mathbf{F}}(\omega) = -\nabla \left\langle \vec{\mathbf{W}} \right\rangle \qquad (20)$$

that is, dielectrophoretic forces direct particles to a regions where their electrical potential energy is minimized.

A positive value of the factor $\operatorname{Re}\left(f(\varepsilon_p^*, \varepsilon_m^*)\right)$ indicates that the particle is more polarizable than its suspending medium. Particles would experience forces termed "positive dielectrophoresis" that direct the particles to regions where the local electric field is maximum. On the other hand, particles which are less polarizable than the suspending medium would correspond to a negative value for $\operatorname{Re}\left(f(\varepsilon_p^*, \varepsilon_m^*)\right)$. In this case the forces direct the particles to a location where the local electric field is minimum and this is referred to as "negative dielectrophoresis". The frequency at which a particle would experience a null DEP force (a value equal to zero for the Clausius-Mossoti factor) is known as the *crossover frequency*. Since the complex permittivity is a function of the frequency, a bioparticle can experience both forces in a range of frequencies.

For example, Figure 2 shows the theoretical frequency dependence (complex permittivity, see equation 17) for particles with conductivity $\sigma_p = 0.3\,S/m$ and permittivity $\varepsilon_p = 10\varepsilon_0$ and a medium with conductivity $\sigma_m = 0.06\,S/m$ and permittivity $\varepsilon_m = 78\varepsilon_0$ (i.e. $\sigma_p\varepsilon_m > \sigma_m\varepsilon_p$) and the theoretical dielectrophoretical (DEP) force and rotational or torque (ROT) spectra of the particles. The average DEP force experienced by the particle is positive up to frequencies of ~10^6 Hz and decays to zero at the crossover frequency of ~ 10^7 Hz. Negative DEP forces act on the particle at frequencies above the crossover frequency. The torque is always positive (co-field rotation) and reaches its peak at the crossover frequency. DEP and ROT (Torque) spectra were normalized as $\vec{\mathbf{F}}(\omega) / \left(2\pi\varepsilon_m R^3 \left|\nabla \vec{\mathbf{E}}_{RMS}^2\right|\right)$ and $\vec{\Gamma}(\omega) / \left(4\pi\varepsilon_m R^3 \vec{\mathbf{E}}_0^2\right)$.

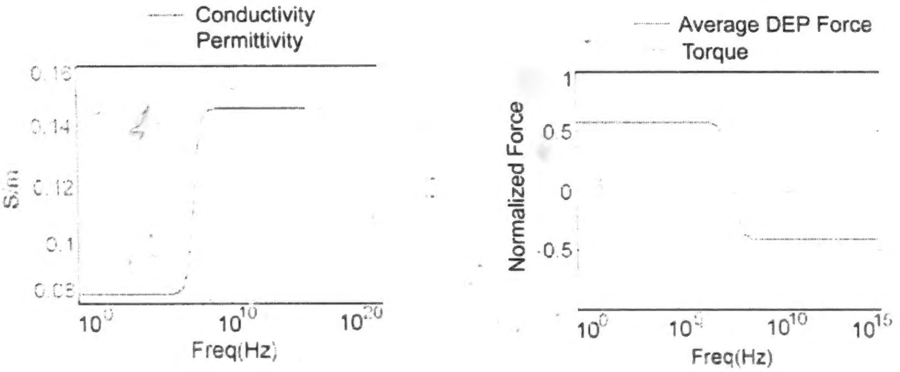


Figure 2. The graph on the left shows the theoretical frequency dependence (complex permittivity, see eq. 17) of particles suspended in a medium that does not exhibit a dielectric dispersion. The graph on the right shows the normalized DEP and ROT responses calculated with the following parameters $\varepsilon_p = 10\varepsilon_0$, $\varepsilon_m = 78\varepsilon_0$, $\sigma_p = 0.3\,S/m$, $\sigma_m = 0.06\,S/m$

Figure 3 shows the theoretical frequency dependence (equation 17) of particles with conductivity $\sigma_p = 0.02\,S/m$ and permittivity $\varepsilon_p = 240\varepsilon_0$ and a medium $\sigma_m = 0.1\,S/m$ and permittivity $\varepsilon_m = 78\varepsilon_0$, where $\sigma_p\varepsilon_m < \sigma_m\varepsilon_p$. The average DEP force experienced by the particle is negative up to frequencies of ~10^6 Hz and increase s to zero at the crossover frequency of ~ 10^7 Hz. Positive DEP forces then acts on the particle at frequencies above the crossover frequency. The torque is always negative (contra-field rotation) and reaches its peak at the crossover frequency.

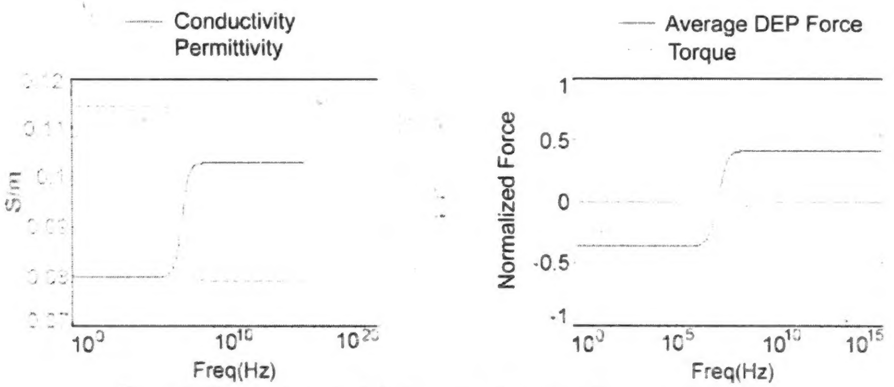


Figure 3. The graph on the left shows the theoretical frequency dependence (complex permittivity, see eq. 17) of particles suspended in a medium that does not exhibit a dielectric dispersion. The graph on the right shows the normalized DEP and ROT responses calculated with the following parameters $\varepsilon_p = 240\varepsilon_0$, $\varepsilon_m = 78\varepsilon_0$, $\sigma_p = 0.02\,S/m$, $\sigma_m = 0.1\,S/m$

Before any dielectrophoretic separation of two type of population of bioparticles takes place, the dielectric properties of the cell and their frequency response need to be known. Determination of physical and electrical properties of particles is accomplished either by measuring their dielectrophoretic induced velocity (DEP velocity) when subjected to a non-uniform alternating electric field. The DEP velocity is proportional to the DEP force (proportional to the square of the voltage applied) [76]. In this case, particles are allowed to reach a stable position (free of any disturbance, no forces acting on them); when electrodes are energized at different frequencies, particles would move to locations where their electrical potential energy is minimized (equation 20, depending on the forces acting on the particles *i.e.* negative or positive DEP). Particle velocity is measured and plotted versus the applied frequency (as shown in Figures 2 and 3) giving the DEP spectrum of the particle.

Obtaining the angular velocity (ROT rate) and sense of rotation when subjected to a rotating electric field is another way to determine the properties of particles. The angular velocity and sense at which particles rotate is a function of the frequency of the applied rotating electric field and is correlated to the DEP spectrum as shown in Figures 2 and 3 [77]. The ROT spectrum is obtained by plotting the angular velocity versus the applied frequency. Measurement and analysis can be performed in single cells [76, 77] or in populations of several hundreds by using computerized imaging techniques enhanced by immunotyping (immuno-fluorescent probes)[28], or optical methods[78, 79].

2.1.3 Heterogeneous Spheres (Cells)

Physicists and engineers have studied the cell following a different approach from that taken by biologists in that they have been able to obtain approximate models of the behavior and responses of the cell to electric and magnetic fields. These electrical and mathematical models have in turn permitted the construction of different devices, such as DEP devices, which are capable of manipulating and characterizing a single cell and multiple cells. It is important then to understand the biological structure of the cell and its chemical composition because the electrical models are based upon them.

Eukaryotic cells are typically approximated as being composed of nucleus, cytoplasm, and membrane, as if they were concentric spheres[2]. The nucleus is very small and is surrounded by the cytoplasm. Their composition is very different: DNA, some proteins and mostly water are found inside the nucleus, whereas for the cytoplasm abundant organelles and more complex molecules are present and its radius is much bigger than the nucleus. The membrane surrounds the cytoplasm in the form of a lipid layer; proteins can be found in this layer[80].

Iramijiri *et al.* [81] proposed a model for the electric properties of lymphoma cells that was based on a series of concentric spheres. Later, this was generalized for the case of eukaryotic cells[76]. The model proposes every shell to exhibit specific frequency-dependant dielectric and conductive properties and assumes that interfaces formed between each shell give rise to dielectric dispersions (relaxation constants). The relaxation time is the time taken to create surface charges at the interface. For example, the thickness of the membrane has been found to measure between 30 Å and 50 Å while a capacitance per unit area of 1 μF/cm^2 and a dielectric constant of 3 was encountered for most plasma membranes [82]. However, their conductance has been found to differ drastically for different type of cells. This dependence arises as the result of the function performed by the cell and of the discrete number of bound-membrane ion channels. The same can be said of the nucleus and the cytoplasm. Those differences are reflected in the values of the in-phase and out-phase permittivities.

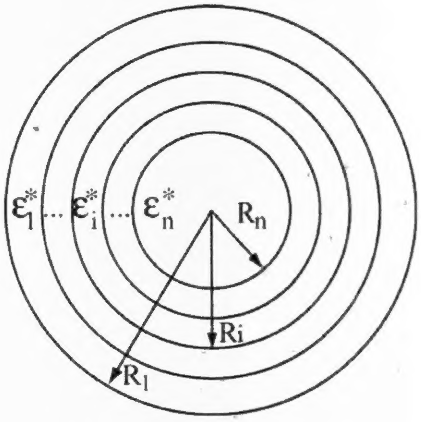

ε_1^* ... ε_i^* ... ε_n^*
R_n
R_i
R_1

Figure 4. Multi-shell model consist of n concentric spheres with different radius and complex dielectric constants

The model is based on the so called "Maxwell-Wagner" interfacial polarization model (*i.e.* the accumulation of charges at the interfaces formed by the shells). It assigns to every concentric shell specific dielectric properties and takes into account the radii; furthermore, it also considers a volume concentration of the particles suspended in a medium with a complex dielectric constant, ε_0^*.

Iramijir *et al.* [81] stated that for a medium with complex dielectric constant ε_0^* and volume concentration Φ of suspended particles with complex dielectric constant $\overline{\varepsilon}_1^*$ and radius R_1, the equivalent homogeneous complex dielectric constant ε^* for the suspension is:

$$\varepsilon^* = \varepsilon_0^* \frac{2(1-\phi)+(1+2\phi)\overline{\varepsilon}_1^* / \varepsilon_0^*}{(2+\phi)+(1-\phi)\overline{\varepsilon}_1^* / \varepsilon_0^*} \tag{21}$$

But $\overline{\varepsilon}_1^*$ can be further subdivided into a series of concentric spheres with complex dielectric constants ε_i^* and radius where i = 1, 2, … , n+1,

$$\overline{\varepsilon}_i^* = \varepsilon_i^* \frac{2(1-\phi_i)+(1+2\phi_i)\overline{\varepsilon}_{i+1}^* / \varepsilon_i^*}{(2+\phi_i)+(1-\phi_i)\overline{\varepsilon}_{i+1}^* / \varepsilon_i^*}, \text{(i = 1, 2, …, n)} \qquad (22)$$

where $\phi_i = (R_{i+1} / R_i)^3$ and $\overline{\varepsilon}_{i+1}^*$ stands for the smeared-out sub phases, that is the innermost shell will be $\overline{\varepsilon}_{i+1}^* = \varepsilon_{i+1}^*$.

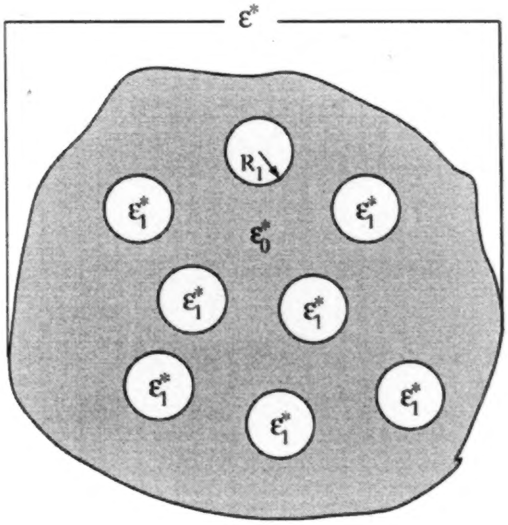


Figure 5. A suspension system with a characteristic dielectric constant ε^* encompasses a medium of ε^*_0 and particles of radius R_1 and dielectric constant ε^*_1

Through a series of substitutions and simplifications we can arrive at the following expression:

$$\varepsilon^* = E + \sum_{k=1}^{n+1} \frac{L_k}{1 + j\omega\tau_k} + \frac{K}{j\omega\varepsilon_0} \qquad (23)$$

where E is the dielectric constant at high frequencies (some authors used the notation ε_∞), K is the conductivity at low frequencies (σ_0), τ_k is the relaxation time of the kth dispersion (or the kth interface), and the associated dielectric increment to this dispersion is denoted as L_k. Equation 23 can then be substituted in equation 16 to include cells modeled in this way.

A single shell model (a particle exhibiting a single dispersion) can be used in the majority of the cases to characterize biological particles [77]. These models have been used to obtain values for the conductivity and permittivity of the nuclear plasma membrane and the nuclear interior of certain cells. Interestingly, the value found for the conductivity of the nuclear membrane 10^4 S/m^2 is two orders of magnitude larger than the plasma membrane on mammalian cells and the value found for the dielectric permittivity is equal to that of water; this can be understood on the basis that a large proportion of ions diffuse across the membrane suggesting the presence of large pores on the nuclear membrane. In fact, those pores with a diameter of 50 nm occupy 1.5 % of the area of the nuclear membrane surface, or 8 pores per µm^2 [77]. An increase in the ionic concentration of the medium would give rise to a decrease in the value of the permittivity of the solution because more ions would be present inside the cell. This understanding can be correlated with the permittivity found for the cytoplasm of some cell groups where a substitution of non-polar molecules for water molecules takes place, and the orientation of water molecules around the solvated ions contributes to this effect, reducing the effective permittivity.

On the other hand, the dielectric polarization mechanism that increases the permittivity of the cytoplasm or other organelles is not well understood, but it is believed that interfacial polarization at the membranes of some organelles, the dielectric relaxation of some type of bio-molecules, or an ionic or protonic displacement along protein structures associated with the cytoskeleton can cause this increment. For example, a DNA concentration of 1% in water exhibits a permittivity of 85 (+7 than the permittivity of water) at 100 MHz [77]. Also, an abundance of amino-acids and small peptide chains increments the permittivity of the cytoplasm proportional to its concentration. However, in some cases, bio-molecules (DNA and proteins) and ions can mutually compensate to yield a permittivity close to 78, the same as water.

In some cases, when for example the nucleus exceeds 30% of the total cell volume or when the volume fraction of vesicles such as mitochondria is considerable, experimental data has to be fitted by double- or triple-shell model (particles exhibiting multiple dispersions) to account for these contributions [77]. An effective model for the case of eukaryotic cells has been found to be the "double-shell model".

The interaction between the cell (and in general a particle) and its surrounding medium introduces an additional interfacial "shell" with its own dielectric properties residing in the suspending medium. This shell may include surface charge, bound ions and a diffuse double layer. Charge double layer effects become dominant for small particles at low conductivities[40]. Electrode surfaces are also susceptible to interfacial effects in the form of electrode polarization. These effects are produced by charge carrier discontinuities between the metal and liquid phases and the development of a contact potential and corresponding charge double layer [11].

In an attempt to probe the validity of the multi-stratified model, Gascoyne, Pehtig *et al.* [83] studied and characterized multi- and oligolamellar liposomes with different diameter to demonstrate the accuracy of this model. Their work concluded that electrorotation must also be accompanied by dielectrophoresis to provide better estimations for the dielectric properties of any bio-particle under consideration.

Other conditions can alter the dielectric properties of some cells. For example, it has been shown that variations in the suspension osmolality (concentration of osmotically active particles in the solution) alter in a dramatic way the conductivity and permittivity of the cytoplasm, and it has been concluded that neither the nucleus nor the mitochondria account for these changes but other organelles must be responsible for these alterations [77].

2.2 Secondary Effects

2.2.1 Pearl Chain Formations

Particles that are more polarizable than the medium distort the electric field surrounding them. Two or more particles close enough would experience the inhomegeneity produced by each of them and mutually attract to regions of higher field intensity near each other. A high concentration of particles then can foster the formation of clusters in the form of "pearl chains" due to dipole-dipole attractions as shown in Figure 6. Pearl chains can hinder the migration of different cell subpopulations and can thus reduce the efficiency of separation methods; some techniques such as switching between frequencies at different intervals and the application of several frequencies simultaneously can help to prevent the formation of pearl chains [2].

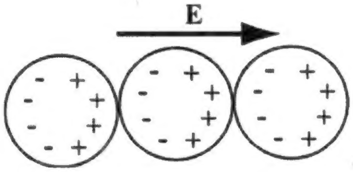


Figure 6. Formation of a pearl chain due to dipole-dipole interactions.

Chapter 3

3 DESIGN OF DEP DEVICES

3.1 Introduction

Dielectrophoresis has found applications in many other fields apart from the separation and characterization of cells applications such as the removal of suspended solids and liquids from non-polar fluids (batch precipitation), the stirring of suspensions (in conjunction with electrophoresis), in the pumping of liquids and powders, and convective cooling[74]. Most of these applications took place using wires or needles as central electrodes and metallic dishes as outer electrodes. Even where they are easy to implement they do not provide sufficient control over individual particulates and the observation of phenomena taking place inside chambers is almost impossible, especially for the separation of cells.

However, microfabrication techniques appeared later on as alternative tools to produce more sophisticated and reproducible electrode structures at the micron scale, mainly through the patterning of metal electrodes on glass substrates. These electrode structures have provided more control over small populations of particles and allow the observation of the phenomena taking place on top of the electrode structure. In addition, microfluidics may allow an improvement in the yield of the separation of particles through the variation of the dimensions of the channels, multiple interconnections, and the inclusion of valves, pumps, and other components. Thus, selection of a proper electrode geometry, electrode size, and microfluidic channel dimensions and geometry are variables that need to be considered when designing a DEP-based microfluidic chip.

Multiple electrode structures and configurations have appeared in the literature. The electrode geometry determines the distribution of the electric field and applications; electrode structure configurations are designed to separate, direct and trap particles at specific locations or to aid in the characterization of different bioparticles. We will discuss in the next section some of the most popular and useful configurations providing an explanation of their advantages, disadvantages and applications.

3.2 DEP Configurations

The tendency in the field has seemed to be limited in choice of electrode structure and has shown a preference for the simplest 2D structures, that is: polynomial electrodes (Figure 7), a series of bus bars (Figure 8) (such as the ones used in traveling-wave dielectrophoresis, explored in Section 3.2.3), inter-digitated electrodes (Figure 9.a), and castellated inter-digitated electrodes (Figure 9.b), perhaps because they are easy to fabricate and offer a relatively straightforward way to characterize and separate cells. On the other hand, three-dimensional (3D) configurations are cumbersome, and present enormous challenges from the engineering point of view. This approach has not been fully developed, but presents a great opportunity for further investigation due to the possibility of 3D electric field cages.

As a first step towards a final DEP-based cell sorting microfluidic device, we decided to explore the simplest configurations. A single mask (see appendix C) contains four different geometries in an equal number of chips. The fabrication procedure is explained in more detail in the next chapter, but briefly, DEP devices consist of two layers. In one layer, gold electrodes were patterned on a glass substrate using standard photolithographic techniques. The size of these electrodes is on the order of 10 to 100 μm. A PDMS layer (see Chapter 4) containing the microfluidic channels was irreversibly bonded to the glass substrate. Two holes were punched at the ends of the microfluidic channels to create inlet and outlet ports. The solution is injected at the inlet and bioparticles are collected at the outlet.

A single chip may contain the same configuration repeated five times but differing only in the variations of the electrode gap. For example, the interdigitated electrode chip contains a set of five electrodes with gaps of 20, 40, 60, 80, and 100 μm. We decided to do this because we wanted to explore the effect of spacing upon particle trapping.

We will explain in the next subsections the advantages and drawbacks of every configuration implemented. Although we did not fabricate quadrupole dielectrophoretic traps[24], we decided to mention it as it seems to be very promising technique for the trapping of bioparticles..

3.2.1 Polynomial Configuration

Positive DEP traps and confines particles at the edges of the electrodes because those are the locations where the electric field is a maximum. Isolated maxima in the magnitudes of three-dimensional, curl and source free electric field cannot occur away from the electrode surfaces [34]. It has been observed that particles collected at the edges of the electrodes can sustain flow rates at least ten times larger than for the same particles trapped by negative DEP at positions away from the electrode structure[84]. Therefore, particles trapped at regions where the electric field intensity is minimum can be more easily flushed away from the electrode structure by fluid flow or gravitational forces.

One configuration used to create isolated electric field minima away from the electrodes is the 'polynomial' electrode geometry [34]. Figure 7.a shows a polynomial electrode structure consisting of four electrodes, although the polynomial configuration can consist of an even number of polynomial electrodes. The DEP force is zero at the centre point (0,0) and increases as $\left(x^2+y^2\right)^{(n-1.5)}$ (where x and y denote the position) up to the edges of the electrodes where is maximum. Particles experiencing negative dielectrophoresis would be directed towards the centre point.

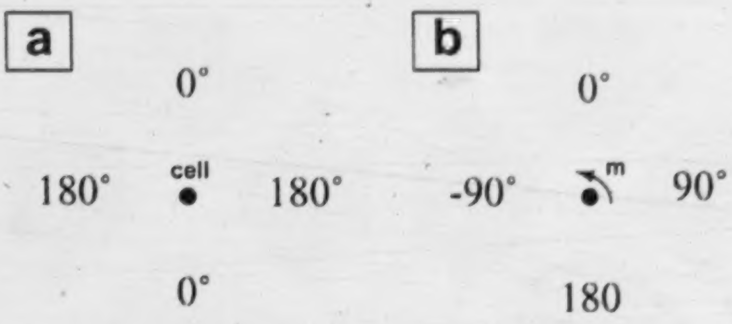


Figure 7. The polynomial electrode configuration can be used to separate cells with the electrodes energized as shown in (a) or to apply a torque (and thus able to obtain the ROT spectrum from a particle) with the electrode setup shown in (b).

This electrode geometry can be either used in dielectrophoretic experiments (signals on adjacent electrodes are phased 180° apart) to separate cells [34, 84] as shown in Figure 7.a or for electrorotation setups (signals on adjacent electrodes are phased 90° apart) to find the rotational spectra [21, 77, 83] of a bioparticle (see section 2.1.2) as shown in Figure 7.b.

Levitation phenomenon has been observed in this configuration. Particles dragged to the center of the electrodes by negative dielectrophoresis levitate as a consequence of the vertical component of the electric force. Particles levitate to a height where the DEP force and sedimentation forces balance each other (assumes electrodes are on the bottom)[31]. This phenomenon has been observed to depend directly on the thickness of the electrodes and the magnitude of the applied voltage [85].

We included this polynomial configuration within the mask (appendix C). The design is based on the references given above; it consists of four electrodes spaced by a gap of 64 μm as shown in Figure 7.

3.2.2 Quadrupole Dielectrophoretic Traps

Voldman *et al.* constructed a microfabricated Dynamic Array Cytometer (μDAC) to trap and hold single cells [24]. The device consist of two rows of electrode posts facing each other, each trap unit is composed of four cylindrical gold electrodes arranged trapezoidally where the adjacent electrodes are powered with alternating polarities. This μDAC seems to be more useful for small population of cells. The holding forces are 100 times stronger than in a planar quadrupole structure (section 3.2.1) under negative DEP force conditions (>100 kHz) so the resistance against high flow rates is increased; working at these high frequencies prevents any electrophoretic effect. Nonetheless, the fabrication process to build the electrodes is elaborate.

3.2.3 Travelling Wave Dielectrophoresis

Travelling wave dielectrophoresis (TWD) produces a linear motion on particles that are less polarizable than the medium. This motion is proportional to the out-of-phase (see section 2.1.2) component of the induced dipole moment and thus is intimately related to the electrorotational force; so negative dielectrophoretical conditions (or negligible positive DEP) must be met in order to be able to produce any motion[21]. The force produced by this configuration is five times smaller than the conventional DEP force. The travelling electric field is produced applying AC voltages of 90° phase separation to a linear array of electrodes (see Figure 8.a). This travelling electric field induces electric multipoles in the particles [86]. The particles interact then with the local field gradients generated by the electrodes and DEP forces as well as electrorotational forces to induce translation and angular motions along the electrodes. Particles travel in the opposite direction to the applied electric field; thus control over the direction where the cell is translating to can be adjusted by reversing the phases of the signals applied (Figure 8.b).

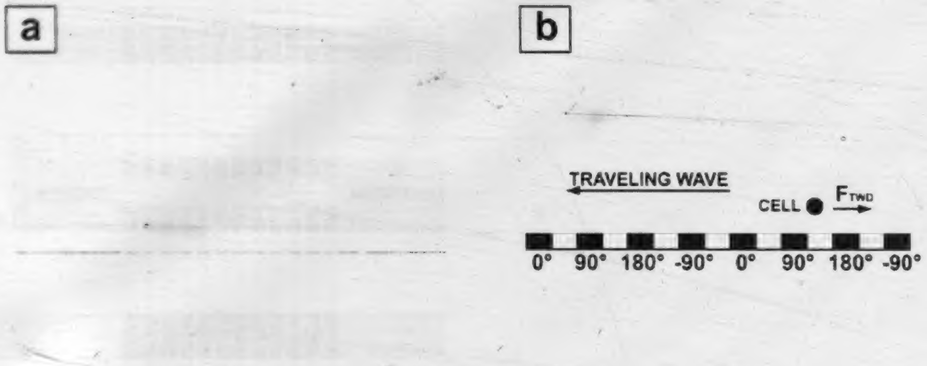


Figure 8. (a) Top view of the comb geometry used to produce travelling electric waves and (b) cross sectional view of the structure showing the direction of the traveling wave. A travelling wave electric field is induced by applying AC voltages of 90° separation. Particles that are less polarizable than the medium translate and rotate in a direction opposite to the travelling wave.

TWD offers advantages such as cell selectivity (size or dielectric properties) and minimization of cell-cell interactions [35]. Travelling wave dielectrophoresis can be used to move around cells in a stationary fluid but lacks precision to position cells because the cell speed is related to the cell type [33]. It is also restricted to a narrow frequency range (<100 KHz). Velocities of different types of cells can overlap as a consequence of biological variability and this has reduced the scope of its application [30]. Furthermore, electrode track lengths of at least 2 cm (or 1000 interdigitated electrode elements) are required to obtain useful and pure fractions [30].

A design consisting of series of bars was included in the electrode mask with gaps of 25 μm between electrodes (see Appendix C), although it was never tested.

3.2.4 Interdigitated Electrodes

Interdigitated electrodes have been used extensively to separate, levitate and trap different types of cells[31]. The configuration is similar to the comb geometry described above but AC voltages are applied. Electrodes can be flat as in Figure 9.a or castellated as shown in Figure 9.b. The DEP force gradient maximum is then located at the edges of the electrodes and decreases exponentially between the electrode gaps and above the electrode area.

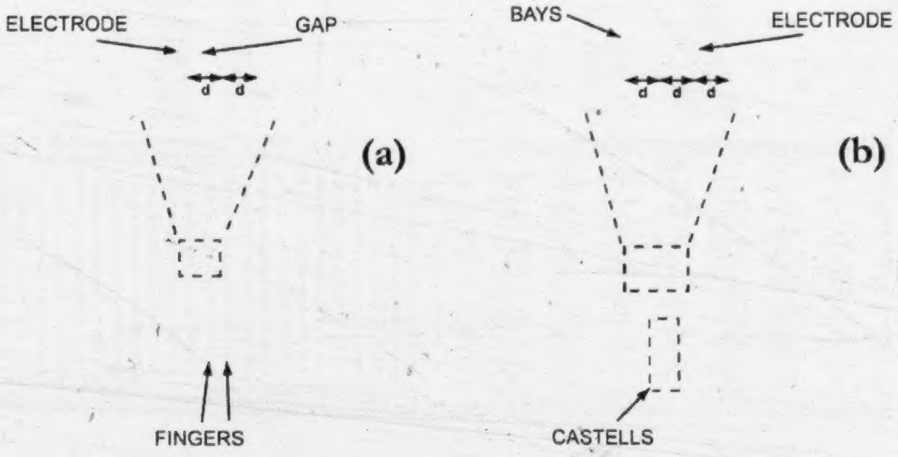


Figure 9. Interdigitated electrode configurations. (a) shows the comb geometry (also called interdigitated electrode structure), electrodes and gaps have the same dimensions. (b) shows the interdigitated-castellated electrode. Castells and fingers are equally separated.

This electrode configuration permits the separation of two populations of cells (Figure 10) by applying an AC voltage at a frequency at with which one type of cells experiences positive DEP while the other population of cells is affected by negative DEP. In this form, cells affected by positive DEP would get trapped very strongly at the edges of the electrodes. The other type of cells experiencing negative DEP are confined very weakly between the electrode gaps or on top of the surfaces of the electrodes. The introduction of a flow would displace the former cells to the outlet of the microfluidic chip (Figure 12)[70]. This configuration has been demonstrated to separate two population of cells by trapping up to 99% of one population of cells and flushing away 90% of the other population of cells[87]. This can be further improved with more DEP stages or other sorting mechanisms.

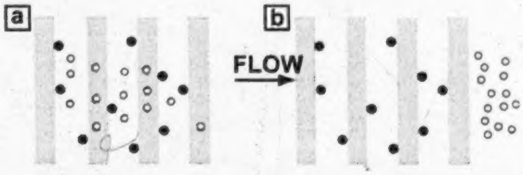


Figure 10. Outline of the separation process of two type of cells using interdigitated electrodes. a) Positive DEP attracts one type of cells (black) to the edges of the electrode, while the other type of cells (white) are collected by negative DEP between the electrode gaps or on top of the electrode surface. b) A stream of flow flushes away the white cells because the negative DEP force is very weak.

This geometry has been used in conjunction with Field Flow Fractionation (FFF)[11]. However, the combination between DEP-FFF leads to a random distribution of cells on the electrode area. This is explained in that cells enter the chamber randomly distributed at the beginning of the separation process, and thus would move with different velocities, which in turn means that cells will experience the positive DEP force for different lengths of time. Moreover, some particles furthest from the electrodes could never be influenced by the DEP force when the length of the channels is small or if the height of the chamber is not designed to account for the exponentially decay of the DEP force [70]. The electric field for this 2D configuration is always maximum at the edges of the electrodes and the magnitude decreases between the electrode gaps and on top of them as shown in section 5.7.1. This could be detrimental in cases where the sample consists of few cells. A group [70] is trying to address this issue using electrodes at the top and bottom parts of the chamber but to our opinion that solution is only a partial solution that greatly increases the complexity of the fabrication.

Interdigitated electrodes can be coupled with traveling wave signals to enhance cell separations and sensitivity (superposition TWD) [30]. They can also be employed to measure the impedance of cells, as sensors, and to form DEP confined cages [33]. Interdigitated electrodes have been used to capture human monocytic leukemia cells (*THP-1*) [70], to separate human T lymphocyte (*Jurkat E6-1*) from monocytes(*U-937*) [30], to trap *Friend murine erythroleukemia DS19* cells [31], to separate malaria-infected cells (*Plasmodium falciparum*) from erythrocytes, to separate and manipulate live and heat-treated cells of *Listeria inocua*[88], to characterize leukemia cells (*Friend murine erythroleukemia clones DS19 and R1*) using computerized image analysis[89], to separate bacteria (heat-killed Eschericia coli or Listeria monocytogenes) from blood [90],

We decided to use interdigitated electrode geometries as one of the strategies for the separation and isolation of cells based on the discussion and arguments given above. Appendix C shows the design of the chips.

3.3 Levitation Effects

Levitation effects always occur for particles experiencing negative dielectrophoresis, especially for 2D plane electrode geometries, where the square of the gradient of the electric field is always pointing to the electrodes, so any particle undergoing negative DEP force would be pushed up against gravity until it counterbalances the sedimentation force. To this end, the forces acting on a bio-particle are given by [31]:

$$2(\rho_c - \rho_m)g = 3\varepsilon_m \operatorname{Re}\left[f\left(\varepsilon_p^*, \varepsilon_m^*\right)\right]\nabla\vec{\mathbf{E}}_{RMS}^2 \bullet \vec{a}_z \tag{27}$$

where ϱ_c and ϱ_m are the specific densities of the cell and the medium, respectively, g is the gravitational acceleration, and $\vec{a}_z$ is the unit vector orthogonal to the plane.

3.4 Considerations for Electrode Sizes

A recent study [91] has shown a limit on the size of interdigitated electrode arrays and to the channel height below which the trapping efficiency decreases for positive dielectrophoresis. The precision to trap particles deteriorates as the electrode width and gap is shrunk because particles aggregate very densely and cover the surface of the electrodes, thus modifying the distribution and intensity of the electric field[91].

The same report suggested that the ratio of the electrode array size to the height of the chamber should not be small [91] because the electric field strength decays exponentially with height. If the chamber volume is too close to the electrodes, particles experiencing dielectrophoretic forces will not be able to move to regions where their electric potential energy is minimized simply because there would not be such regions. The worst scenario would happen when the height of the chamber has the same dimensions as the diameter of a particle.

3.5 Simulations

We built and tested different electrode configurations. Chapter 5 describes the experiments realized on different electrode geometries; each configuration is accompanied by its own simulation. Simulations for the different electrode configurations were performed to aid in the understanding of the distribution of the electric field intensity.

Simulations for all structures were performed in FEMLAB v.3 (COMSOL, USA). Three-dimensional models were created using the 3D Electrostatic Generalized application mode found in the Electromagnetism Module. Solutions were computed using GMRES as the linear solver and Algebraic Multigrid as the preconditioner. An example of the ".m"output file generated by FEMLAB used to simulate the electric field distribution for the quadrupole electrode configuration (section 5.4.1) is given in Appendix H. The ".fl" output files generated by FEMLAB for all the simulations were stored in a CD. The location of the files can be found in Appendix I.

All models consisted of a cubic chamber and electrode structures placed on the bottom of the chamber. Chambers enclosing the electrodes had a height of 50 microns and its walls as well as the bottom surface were modeled as electric insulators (susceptibility equal to zero). Electrodes were modeled as perfect electric conductors (infinite conductivity) with thicknesses of 5 microns to facilitate and reduce the time needed to generate the mesh topologies. Electrode models have the same dimensions as the constructed electrodes. Water (relative permittivity, 72) was used to simulate the medium. Electric potentials for the electrodes were set at 5 and -5 Volts DC in most of the cases.

Chapter 4

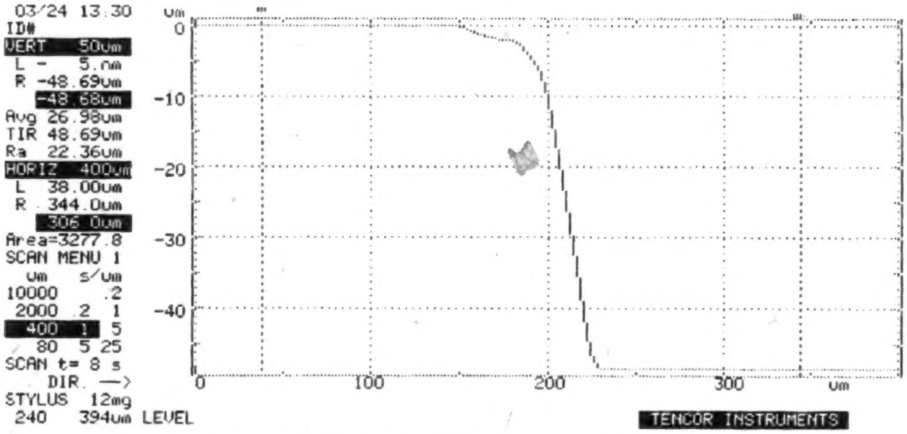
4 FABRICATION

4.1 PDMS Master Mold Fabrication

Chip designs were drawn in L-Edit ver 8.0 (MEMS Pro, MEMS CAP, Cedex, France). A pattern generator (DWL 200, Heidelberg Instruments, Heidelberg, Germany) was used to transfer the design to a chromium mask. A 4"x4" borofloat glass substrate coated with chrome and gold was purchased from Micralyne (Edmonton, AB, Canada). The substrate was rinsed three times with acetone, isopropanol (IPA), and deionized water and blow-dried under a stream of N_2. Next, although Hexamethyldisilazane (HMDS) is commonly used to improve photoresist adhesion to oxides, the substrate was placed in a HMDS oven (Yield Engineering Systems, San Jose, CA, USA) in the hope of enhancing photoresist adhesion. The HMDS oven was heated up to 150 °C and pulled to vacuum until the pressure in the chamber read 1 Torr. The substrate was then exposed for 10 seconds to HMDS vapor. The oven returned to atmospheric pressure and the substrate was extracted.

The substrate was then placed on a spinner (#5110-CD, Solitec Spinner, CA, USA) configured with the following parameters: spread speed 200 rpm for 13 seconds (cycle 1), spin speed 1200 rpm for 15 seconds (cycle 2). Five milliliters of SRJ5740 ultra-thick photoresist (Shipley Microelectronics, MA, USA) were manually poured on the substrate while it was spinning (cycle 1). The substrate was left atop the chuck for 1 min. Next, a soft-bake step in a preheated oven to 115°C for 30 minutes was performed. The substrate was stored overnight in a box containing a 10 ml beaker of water. The photoresist coating thickness produced was about 25 microns. We were able to obtain 48 micron features by double coating a substrate as shown in Figure 13. (We repeated the procedure with the same parameters to obtain thicker coatings.)

UV exposure (356 nm) was achieved using a contact mask aligner system (AB-M Inc., Silicon Valley, CA, USA). The mask was brought into contact with the substrate and exposed for 40 seconds (λ=365nm, 18.3 mW/cm^2). Following this, Developer 354 (Shipley Microelectronics, MA, USA) was poured in a container and the wafer immersed in it; the container was manually agitated for 45 minutes (endpoint was determined by visual inspection). Wafers are then rinsed with water. Photoresist relief thickness was measured with an Alphastep 200 Contact Profilometer (KLA-Tencor, San Jose, CA, USA) Figure 11. We obtained 5 replicas from the same mold, although master molds can used to fabricate more than 50 PDMS replicas[92].



```
03/24 13:30
ID#
VERT    50um
 L -    5.nm
 R -48.69um
   -48.68um
Avg 26.98um
TIR 48.69um
Ra  22.36um
HORIZ   400um
 L  38.00um
 R  344.0um
    306.0um
Area=3277.8
SCAN MENU 1
  um    s/um
10000     .2
 2000  .2  1
  400   1  5
   80   5 25
SCAN t= 8 s
    DIR. —>
STYLUS  12mg
 240   394um LEVEL
```

Figure 11. Substrates double-coated with SJR7540 photoresist were characterized with an Alphastep profilometer. Measured value was 48.68 μm.

4.1.1 Interconnections

The operation of microfluidic systems relies on the creation and control of a stream inside the channels. There are multiple ways to pump a fluid within a microfluidic chip as reviewed in chapter 1; since each of them is a field of research on its own, it is not within the scope of this thesis to explore or employ all of them, however, one of them had to be chosen. We decided to employ the simplest methods to pump our solution in the channels: applying vacuum to one of the wells [19] or by applying positive pressure to the inlet ports and leaving the outlet ports exposed to atmospheric pressure (pressure-driven).

Still in its infancy, microfluidic systems are ultimately expected to be fully independent and operate without the aid of any macroscopic tool such as syringe pumps, pipettes, or microscopes. To this end, we adopted simple fabrication procedure to create the interfaces between the external fluidic network and the microfluidic chips.

The interconnections between PDMS-based microfluidic devices and external fluidic networks pose several engineering challenges. Trying to connect syringes via capillary tubing to the inlet port of a microfluidic chip while preventing the formation of dead volumes is one of them. Leakages are prone to appear around the ports if the interconnection has not been properly sealed, if the punched hole is not circular, or if the pressure of the fluid is too high. High pressures can also detach the PDMS from the substrate of a once irreversibly-bonded chip (see section 4.3). Clogs due to the same sample or dust are commonly found. Compatibility between materials, solutions and particles is another issue to be taken into account. In some cases, alignment of inlet and outlet ports with macro can increase the complexity of the design. Many papers have been published to tackle these issues using different approaches [93-98].

Stereolitography has been lately used to create micro to macro fluidic interfaces [93]. The technique uses a photosensitive polymer and a laser beam to pattern the desired structure layer by layer. The minimum feature size that can be created with this technique is 75 μm and provides a vertical resolution of 50 μm [99]. Despite the several advantages offered by stereolitography to create intricate designs, the process is generally expensive, and depending on the complexity of the design, it can take up to several hours to create a design.

Ports of the microfluidic chip can be located either on the PDMS layer or in the substrate, *i.e.* atop or underneath the chip [93-95, 97], or as well, from the sides [98, 100]. Placing ports and interfaces along the sides of the chip is laborious, requires extra steps in the fabrication process[98], it is time-consuming, and it does not seem to be a popular technique. Interconnections located at both surfaces of the chip appear to be simpler and easy-to-fabricate, their success seems to depend more on the type of experiment, and on the parameters or conditions required.

One approach to define access ports at the surface of the microfluidic chips is to drill the access ports on the glass substrate. But special port adapters (fittings, ferrules, or sleeves) [101] are required to interconnect the tubing with the glass due to the rigidity of the latter, increasing the cost of the device. Metal or glass capillary punches[95] and hypodermic syringe needles[94] have been used to define access ports on the PDMS layer or other soft material. However, punches and needles have to match the exact dimensions of the tubing that would be attached to in order to prevent leakages. Epoxy resins can be used to firmly attach the tubing to the ports if the size does not match [94, 95].

The fabrication approach that was adopted for this thesis is described in the next section.

4.1.2 PDMS Replicas

A squared metal holder built-in-house was used to contain the master mold. Ten grams of PDMS pre-polymer and its curing agent (Sylgard 184, Dow Corning, MI, USA) were mixed and stirred thoroughly in a ratio of 10:1 (25:2.5 grams). It was degassed in a vacuum chamber/oven (model #1415M, Sheldon Lab, OR, USA) for 30 minutes at a pressure of 25 cmHg. The solution was then poured over the master mold to a thickness of 3mm.

PEEK Tubing Sleeves (.031" ID, 1/16" OD, F-232, Upchurch Scientific, WA, USA) were chopped up to a length of 4 mm and 8 mm; the cut had to be as flat and smooth as possible and for this purpose a Capillary Polymer Tubing Cutter (A-350, Upchurch Scientific, WA, USA) was used. Four millimeter long tubing pieces were then placed manually on top of the PDMS layer to define what will be future inlet and outlet ports, see Figure 12.e. PDMS penetrated inside the capillary tubes and held the posts in a vertical position. This was possible because the consistency of PDMS is very viscous. However, if the cut is not completely flat, tubes can tip over.

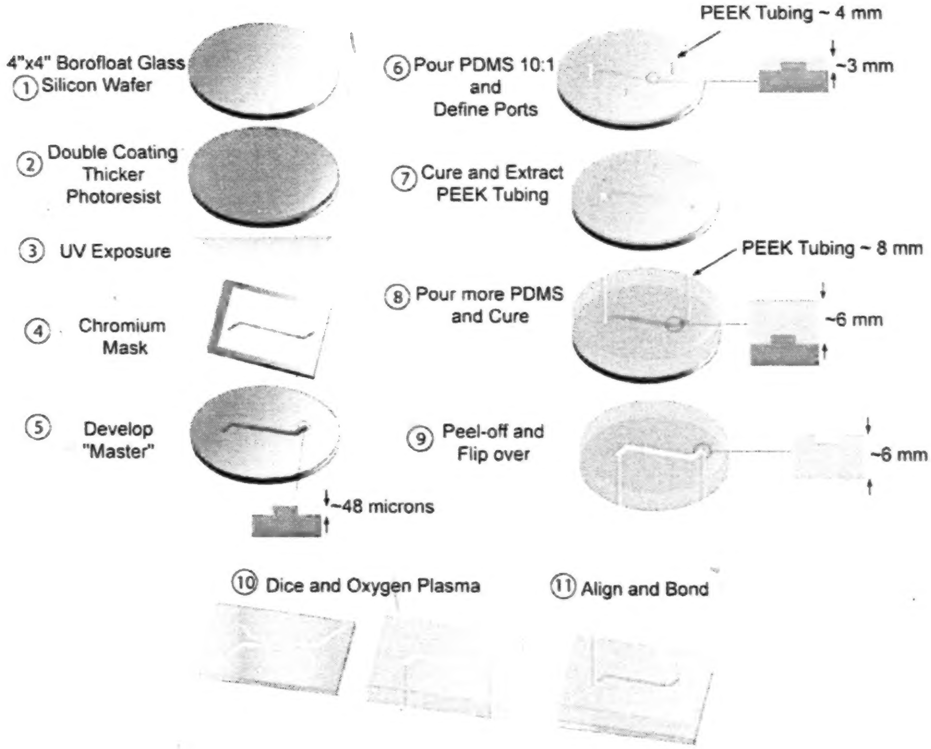

4"x4" Borofloat Glass
① Silicon Wafer
② Double Coating Thicker Photoresist
③ UV Exposure
④ Chromium Mask
⑤ Develop "Master"
~48 microns
⑥ Pour PDMS 10:1 and Define Ports
PEEK Tubing ~ 4 mm
~3 mm
⑦ Cure and Extract PEEK Tubing
PEEK Tubing ~ 8 mm
⑧ Pour more PDMS and Cure
~6 mm
⑨ Peel-off and Flip over
~6 mm
⑩ Dice and Oxygen Plasma
⑪ Align and Bond

Figure 12. Schematic procedure for the creation of the master mold and for casting PDMS with PEEK posts serving as ports

Next, PDMS was cured inside a PDMS-dedicated oven (Model 1415M, Sheldon Manufacturing Inc., OR, USA) at 80° C for one hour. Half an hour later, once it had cooled down, it was taken out. Tubing sleeves were extracted from the PDMS replica and they were replaced by 8 mm long tubing sleeves. PDMS elastomer prepared (as described before) was poured over the replica to create an extra layer and cured for the same time at 80° C. The purpose of this layer is to give mechanical support to the longer sleeves. The PDMS replica was peeled off from the master mold and bonded to a glass substrate as will be explained in the next section (Figure 13).

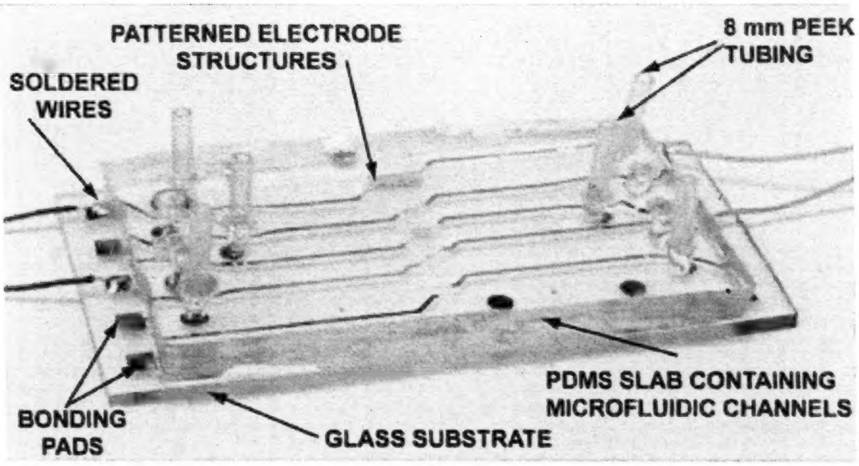

PATTERNED ELECTRODE STRUCTURES
8 mm PEEK TUBING
SOLDERED WIRES
PDMS SLAB CONTAINING MICROFLUIDIC CHANNELS
BONDING PADS
GLASS SUBSTRATE

Figure 13: DEP Microfluidic chip with ports showing the different parts composing it.

In a few of the cases, the ports were found to be clogged by PDMS because the tubing sleeves were not cut completely flat and thus they could not be positioned completely orthogonal to the master mold. However, a punch can be used to unblock the clogged ports at the risk of leakages once the sleeves are put back.

A 1/16" x 3/32" reducing miniature barbed polypropylene fitting (model 6365-90, Cole-Parmer Instrument Co., IL, USA) was used to couple the ports with the fluidic network as shown in Figure 14. Dead volumes or leakages were not observed using this configuration. Although not very rigid, the ports can withstand flow rates up to 200 µl/min without leaking.

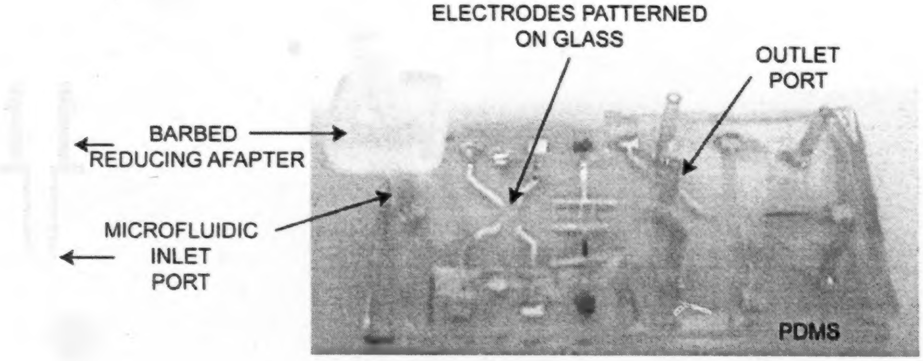

ELECTRODES PATTERNED ON GLASS
OUTLET PORT
BARBED REDUCING AFAPTER
MICROFLUIDIC INLET PORT
PDMS

Figure 14. Schematic showing the adapter used to link the external fluidic network and the ports embedded in the chip (left). Picture showing the chip with one adapter connected to the inlet port (right).

4.1.3 Discussion

Interconnecting microchannels in this way is rather easy; however there are still some issues to be solved. The main concern is the transition from the external fluidic network (inner diameter 1.58 mm giving a cross sectional area of 2 x10^6 μm^2) to microfluidic channels (the cross section areas were in the order of 1 x 10^4 μm^2 in our case). This difference in cross sectional areas can give rise to problems such as: higher pressures at the interconnection point and variations in the fluid velocity within the channel. Furthermore, a high concentration of bioparticles in a solution can possibly clog the interface between the syringe and the microfluidic channel but we found that this could be avoided by using diluted solutions.

4.2 Electrode Fabrication

Different electrode configurations were patterned on a 4" x 4" Borofloat wafer coated with gold (thickness: 200 nm) on chrome (thickness: 20 nm) (Micralyne, AB, Canada). Chip designs and chrome masks were designed and fabricated as described in Section 4.1.4. The substrates were spin coated with HPR504 photoresist (Shipley Microelectronics, MA, USA) at a spread speed of 400 rpm for 10 seconds and at a spin speed of 5000 rpm for 12 seconds. It was then soft-baked for 30 minutes at 115 °C and cooled for 15 minutes. The substrate was UV exposed (λ=365nm, 18.3 mW/cm^2) for 4 seconds in a contact mask aligner system (AB-M Inc., Silicon Valley, CA, USA); unexposed photoresist was stripped using Developer 354. Gold was etched with a gold etchant (100 g Potassium Iodide and 25 g Iodine dissolved in one liter of deionized water) for about 50 seconds (the end point was determined by visual inspection). The substrate was rinsed with deionized water and blow-dried with a Nitrogen gun. Next, the chrome layer is etched away using a chrome etchant (Arch Microelectronics Materials Inc, CT, USA) for about 20 seconds; end point determined by visual inspection. Fabricated electrodes are shown in Figure 15.

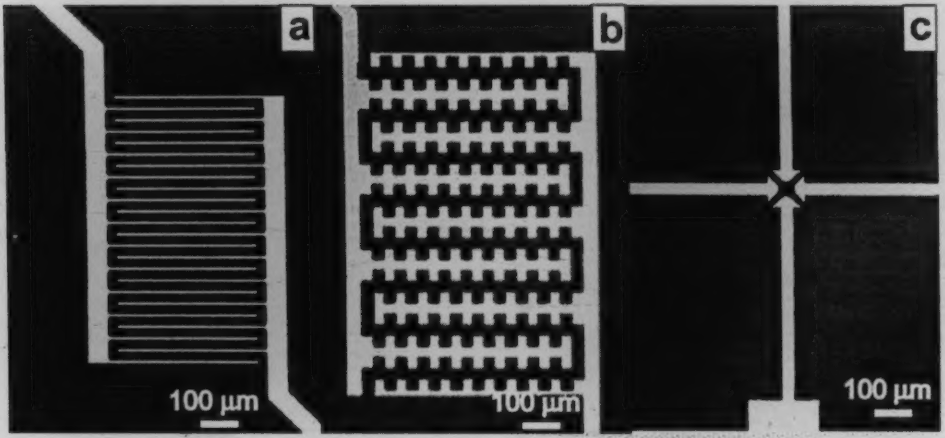


Figure 15. Snapshots of (a) interdigitated electrode configuration, (b) interdigitated castellated electrode configuration, and (c) polynomial electrode configuration.

4.3 Bonding

4.3.1 Introduction

PDMS can be irreversibly bonded to PDMS, glass, silicon, polystyrene, polyethylene, silicon dioxide, quartz, glassy carbon, and silicon nitride[47]. It can reversible seal to any smooth surface due to conformal contact (van der Waals bonds), PDMS and glass being the most common substrates[48].

Reversible bonding is more appropriate for be use in applications such as capillary electrophoresis, for patterning surfaces with proteins and cells, or for low-pressure (<5 psi) applications. Cleanness is facilitated because PDMS can be demounted, washed, and wiped off if required, as many times as desired without degradation in the PDMS [47]. If glass is used as a substrate, Piranha (Sulphuric Acid : Hydrogen Peroxide in a ratio 1:3 by volume) can dissolve any organic contaminants that could have stuck to the glass.

Irreversible bonding is more suitable for high-pressure (30-50 psi) applications [47]. To this end, a protocol to irreversibly seal the PDMS replicas to glass substrates using an oxygen plasma exposure was developed.

4.3.2 Procedure

PDMS replicas were sprayed with ethanol for about 5 seconds and dried immediately under a stream of nitrogen. Glass substrates with patterned electrodes were sprayed with acetone, IPA, deionized water, and finally blow-dried with a N_2 gun. Both pieces were placed in a Reactive Ion Etcher (µetch-RIE, Tegal Corp., CA, USA) with the surfaces to be bonded facing up. They were oxidized for 1 minutes 45 seconds (80% O_2, 0.25 Torrs, 40 Watts). The PDMS replica and the glass substrate were immediately stored in a Petri dish to prevent any contamination. The two pieces were carried to the photolithography room. The pieces were aligned in the contact mask aligner and brought into contact (see next section); this procedure took approximately two minutes. Next, the microfluidic chip was placed in a hot plate (Isotemp 11-600-49SH, Fisher Scientific, ON, CA) with the temperature set at 80 °C for five minutes. It was left to cool down for 10 minutes and taken again to the RIE with the same settings but this time for 2 minutes. A post-oxidization treatment was found to enhance the strength of the sealing (the adhesion of PDMS to glass). Irreversibly bonded microfluidic chips can withstand pressures up to 100 psi[102]. Finally, the microfluidic chip was stored in a box.

4.3.3 Alignment

In some designs a precise alignment between the PDMS replica and the patterned electrodes on glass is necessary. Using tweezers or the hands to procure an exact alignment is rather cumbersome and almost never accomplished. We found it easier to use the mask aligner to line up both. We describe the procedure followed to align the PDMS layer containing the microfluidic channels on top of the glass substrate. Although, we could not obtain an exact alignment, we were capable of aligning the PDMS layers to within 20 microns.

After oxidation of both substrates, the glass substrate containing the electrode was taped (Highland Transparent Tape 5910, 1/2 inch, 3M, MN, USA) to a microscope slide and the PDMS layer containing the fluidic channels adhered to a blank square glass substrate (same dimensions as the chrome mask); the hydrophilic nature of the just oxidized PDMS bonds it reversibly to the substrate. The microscope slide is placed on the chuck of the mask aligner (AB-M Inc., Silicon Valley, CA, USA) and the blank glass substrate is put in the mechanical stage designated for chrome masks. Vacuum is applied to the chuck and the mechanical stage. The mechanical stage is brought down. Knobs are used to move the stage containing the chuck and thus align it.

Next, surfaces are brought into contact by lifting up the chuck. The PDMS layer then adhered to the patterned glass substrate. Finally, the chuck vacuum was released and the mechanical stage lifted up. The PDMS adhered to the unpatterned glass substrate was detached from the blank square glass substrate by just pulling it, see Figure 16. The bonding procedure continued normally as described in Section 4.3.2.

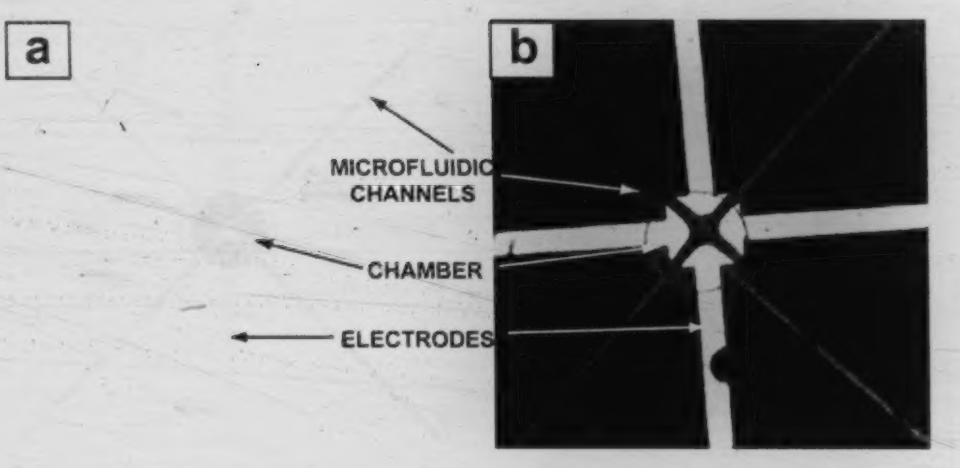

a
b
MICROFLUIDIC CHANNELS
CHAMBER
ELECTRODES

Figure 16. Alignment of polynomial electrodes and PDMS channels. Figure to the left shows the original design and the expected alignment. The width of the PDMS microfluidic channels is 20 microns. Figure to the right shows the built chip, PDMS layer containing microfluidic channels were aligned and bonded to the glass substrate with an accuracy of 20 microns.

Chapter 5

5 TESTING OF DEP DEVICES

Before embarking in the description of the different experiments performed in the microfluidic chips, we will start out this chapter dedicating the first section to the treatment and conditioning of PDMS microfluidic channels. Conditioning of microfluidic channels prior to any assay is regarded as crucial to the outcome of any experiment because of the surface properties of PDMS. Later, we give a description of the experimental setup to conclude the chapter with sections describing the experiments performed in different electrode structures using white sulfate beads (diameter 9 μm) and surfactant-free red carboxylate-modified latex beads (diameter 4 μm). Beads were obtained from Interfacial Dynamics Corporation (OR, USA). A simulation of the electric field accompanies each subsection as well as a description

5.1 Effects of PDMS Channels

There is one property that has precluded the use of PDMS in more applications up to now, or at least has made it more difficult to integrate: PDMS is hydrophobic by nature (because of its low surface energy) and thus is difficult to wet and prone to air bubble formation [47, 103]. Non-specific or non-desired proteins can absorb into the channels because of this characteristic and dramatically affect measurements [104, 105]. Furthermore, depending on the environmental conditions, PDMS can swell, shrink, or sag, which can limit feature geometries [19, 106]. We encountered these problems when performing our first experiments.

5.1.1 Experimental

We fabricated the first microfluidic chips as described in Chapter 4 and our first experiments included the separation of latex beads. The solution was prepared by mixing 10 μl of solid stock solution of white latex beads (collected with a 20 μl pipette) and poured in a 10 ml plastic tube containing 1 ml of deionized water. We injected this solution into the microfluidic chips and realized that after few seconds the beads (as well as yeast cells) bound strongly to the surface of the channels (glass and PDMS) because of hydrophobicity. We also observed that the DEP force was not sufficient to overcome this force in order to release the beads.

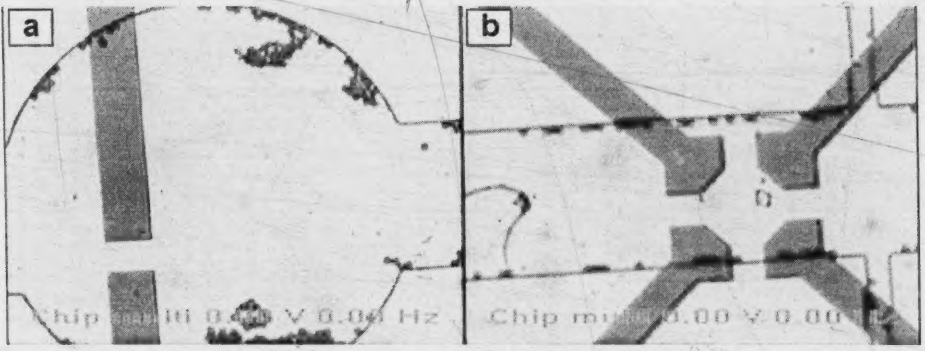


Figure 17. Beads have adhered to the walls and bottom surface of the chamber as a result of hydrophobicity. Pictures were taken with a video camera (VK-C350, Hitachi, Tokyo, Japan) attached to an Olympus microscope (BH2-UMA, Olympus, NY, USA)

Different solutions (sulfuric acid, acetone, and ethanol) were used in the hope of removing the beads and cleaning the channels without success. A sonication treatment in water for 30 minutes could release most of the beads and cells attached to the surface of the channel. Although we immersed the chip in a clean solution within a clean container, we believe that contamination issues would arise because the outer surface of the chip was in direct contact with the external world when performing the experiments, and thus contaminate the cleaning solution.

We also observed the formation of air bubbles occasioned by the hydrophobic nature of the PDMS surface as shown in Figures 18.a and b. The advancing contact angle of water on PDMS is 108°; thus, aqueous solutions have a tendency to dewet from the surface of the hydrophobic surface of the PDMS [103].

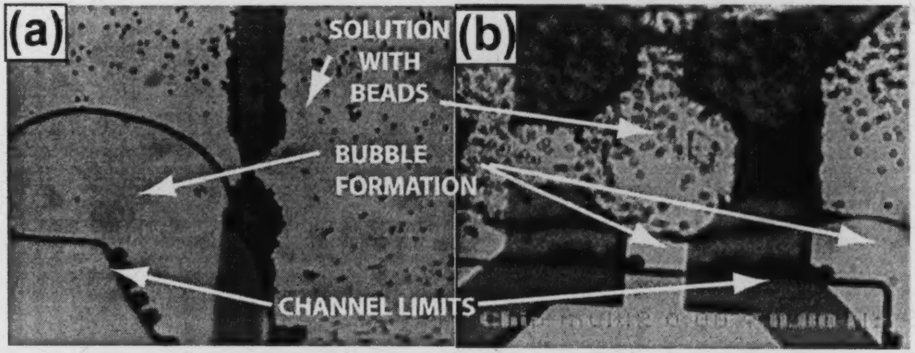


Figure 18. Bubble formation inside a microfluidic chip

5.1.2 Treatment of Fluidic Channels

There are some methods that can render the PDMS surface hydrophilic. Some of the most common are discussed below, other more complex methods such as using polyelectrolyte multilayers, radiation-induced graft polymerization, cerium (IV) catalyzed polymerization, chemical vapor deposition, and phospholipid bilayer modification are discussed in more detail in the literature [107].

The most common method is to expose PDMS to an oxygen plasma discharge so it converts $-OSi(CH_3)_2O-$ (the CH_3 groups that make its surface hydrophobic) present at the surface into polar groups . It is under discussion which chemical groups appear at the surface once it has been treated[107-109] (probably $-O_nSi(OH)_{4-n}$ [103], silanol groups[48]), but the effect lasts for approximately three hours if it is left exposed to air [103, 108], and can last up to three days if it is submerged in water[103]. However, this procedure is time-consuming and can cause contamination issues.

Another method used to decrease the hydrophobicity of PDMS channels is to condition the channels with 0.1 M NaOH for at least 60 minutes, washed with deionized water for 3 minutes, and flushed 10 minutes with buffer, it is also recommended to use acidic solutions rather than basic solutions to prevent the formation of air bubbles [109]. An alternative is to initially wet the PDMS channels with a non-ionic surfactant (Triton X-100) and use a 1% solution of bovine albumin to reduce enzyme adhesion [110]. Although both groups reported its effectiveness to reduce hydrophobicity based on observation, neither group suggested the operative mechanism.

A group from Japan reported two alternative methods, soaking the PDMS in 1M HCl and immobilizing γ-aminopropyltriethoxysilane through silane coupling reaction after exposure to oxygen plasma, although they did not explain the mechanisms involved[111].

One more method involves either the treatment of channels with proteins (bovine serum albumin - BSA) prior to any experiment or mixing the assay solution with different reagents so hydrophobicity does not affect or inhibit to a great extent the experiments as demonstrated in PCR assays performed on microfluidic chip [104, 105]. It has been found that one of the causes that may lead to inefficiency in on-chip PCR amplifications is that the reagents (specially the Taq polymerase) absorb to the chip walls or chambers, and this effect is accentuated by the large surface-to-volume ratios encountered in these micro-systems[105]. To counteract this problem, passivation reagents such as a protein adjuvant (BSA) [105, 112, 113] or polymers (PEG, PVP) [104] can be introduced to compete with reagents for active adsorption sites at chip walls and prevent components of the sample from binding to the walls. This can be accomplished by treating the walls with the passivation reagent before the assay takes place (static passivation) or by including the passivation reagent in the reaction mixture (dynamic passivation) [104]. Ethanol or methanol can be used after the measurement to clean up the chips.

We adopted this final method to condition the microfluidic channels. BSA in a proportion of 10 mg/ml was used as an adjuvant to treat the surface of the channels. Its preparation and protocol of operation can be found in Appendix A.

5.2 Experimental Setup

DEP devices need an instrument to energize the electrodes and either a syringe or vacuum to drag the solution through the microfluidic channel. Figure 19 shows the setup we used to test the DEP devices. Alternating electric fields are generated via a function signal generator (HP33120A, Hewlett-Packard, CA, USA) and signals are connected to the bonding pads on the microfluidic device. The solution can be injected to the inlet of the chip using a syringe or by loading the inlet with a pipette and letting the vacuum connected at the other end to drag the solution through the channel. A syringe pump (KDS220, KD Scientific Inc, USA) was used when needed to inject different solutions into the microfluidic chips.

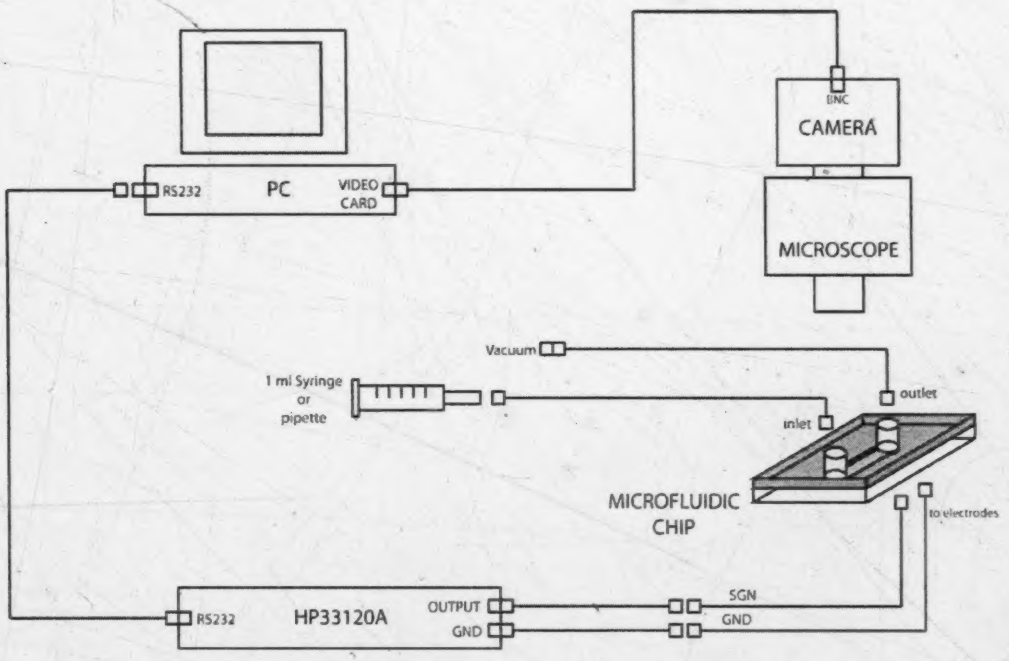

RS232
PC
VIDEO CARD
BNC
CAMERA
MICROSCOPE
Vacuum
1 ml Syringe or pipette
outlet
inlet
MICROFLUIDIC CHIP
to electrodes
SGN
GND
RS232
HP33120A
OUTPUT
GND

Figure 19. Experimental setup for DEP devices. Details of the instruments are found in the text.

The function signal generator is capable of supplying sinusoidal signals up to 15 MHz and 10 Vpp with a 50 ohm load or 20 Vpp with a 1 Megaohm load (High impedance). BNC cables with a 75 ohm characteristic impedance were used to convey the signals to the microfluidic chip. To match the impedance of the microfluidic chips, 75 ohms or 1 Megaohm resistors had to be connected in parallel with the function generator. Signals are monitored with an oscilloscope HP54600A (Hewlett-Packard, CA, USA). The function generator can be controlled from a computer through the serial port. A camera attached to a microscope and connected to the computer is used to observe and record the experiments.

Experiments were observed using an Olympus microscope (BH2-UMA, Olympus, NY, USA). Videos were captured through a solid state color video camera (VK-C350, Hitachi, Tokyo, Japan) attached to the microscope. A video capture card (ATI TV Wonder Capture Card, ATI Technologies, ON, Canada) converted analog signals coming from the camera to a digital format.

A built-in-house software program developed in Visual C++ (Microsoft, WA, USA) and based on DirectX 8.0 technology (Microsoft, WA, USA) was used to store video digitally in a PC station (Pentium 4 Processor, Intel, CA, USA). The program also permitted the acquisition of still images (every 100 milliseconds) for further analysis.

5.3 Materials and Methods

5.3.1 Beads

Latex beads were used in the first experiments due to the homogeneity of their size, material and surface charge. White sulfate beads with a diameter of 9 µm (solid content: 4.2 % by weight in water) and surfactant free red carboxylate-modified latex (CML) beads with a diameter of 4 µm (solid content: 2.3% by weight in water) were purchased from Interfacial Dynamics Corporation (Portland, OR, USA).

CML beads are hydrophilic but when the pH of the buffer that contains them is greater than 5, charged groups are neutralized and CML beads form aggregates. In addition, the surface of the sulfate beads is hydrophobic and binds strongly to any other hydrophobic material or molecules such as proteins and nucleic acids[114].

To reduce this effect, the manufacturer recommends coating them with BSA, dextrans, or through specific buffer pHs in the case of CML beads (which complicate the preparation of the solutions at specific conductivities). Such buffers have high conductivities due to high concentration of salts.

BSA was used as a blocking agent to prevent any adhesion to the surface of the channels. The solution was prepared by mixing 4 µl of solid stock solution of beads (previously vortexed) with 10 µl BSA (10 mg/ml) in a 20 µl vial, then vortexed, and finally suspended in a 100 ml plastic tube containing 20 ml of deionized water. Solutions were stored in a refrigerator at 4°C for up to one month. Conductivity, measured with a Thermo Orion model 150 Conductivity Meter (Thermo Electron Corp., MA, USA), was found to be 3 µS/cm.

5.3.2 Buffer Solutions

A 0.1 M phosphate buffer solution with pH 7 was prepared by taking 0.7 g of Na_2HPO_4 plus 0.5 g KH_2PO_4 and 0.5L of deionized water, and stirring thoroughly. The conductivity was 160 μS/cm as measured by a Thermo Orion Model 150 Conductivity Meter (Thermo Electron Corp., MA, USA).

5.3.3 Cleaning Procedure

Channels were flushed with methanol for 10 minutes, rinsed with buffer solution for 10 minutes, methanol again for 10 minutes, and deionized water for 10 minutes. Finally, they were dried under a stream of nitrogen (nitrogen is an inert gas).

5.3.4 Channel Conditioning

Microfluidic channels were first loaded with methanol (a polar organic solvent) for 10 minutes to facilitate the introduction of aqueous solutions and to prevent bubble formations[109]. Phosphate buffer was then run through the channel for 5 minutes followed by deionized water for another 5 minutes. Microfluidic channels were then filled with 10 mg/ml of BSA (BSA 96% Electrophoresis grade, A2153-10A, Sigma-Aldrich, MO, USA), as a blocking agent prior to the introduction of the samples. It was left for half an hour, although some papers reported 2-3 minutes on a 2.5 mg/ml on PCR chips based on glass[113] as adequate, we adopted half an hour as an standard procedure. Lastly, deionized water was flushed through the channels and the microfluidic chip was ready to be used.

5.4 Quadrupole Electrode Structure

We studied the quadrupole electrode structure using white and red latex beads exhibiting different properties. Hydrophobic white latex beads (diameter 9 μm) contain sulfate groups at their surface (sometimes hydroxyl and carboxyl groups are present) and have a surface charge density of one charged group for a surface area ranging from 200 $Å^2$ to 2000 $Å^2$ and [114]. Hydrophilic red latex beads (diameter 4 μm) have a highly charged surface (produced by copolymerizing carboxylic acids containing polymers) and a surface charge density of one charged group for a surface area ranging from 10 $Å^2$ to 100 $Å^2$. We analyzed the effects of this electrode configuration on both particles at different frequencies and found the approximate crossover frequencies for both beads.

5.4.1 Simulation

We simulated this structure as described in Chapter 3 to get an idea of the electric field strength distribution. Figure 20 shows the result of the simulation where electrodes a) and b) were set at 5V and -5V respectively, and electrodes c) and d) were set at ground. We can observe that the electric field is stronger between electrodes a) and b) and thus we will expect particles to collect at this region when they experience positive DEP forces. The electric field decreases as we move further from the electrodes regions where the electric field vanishes. Particles experiencing negative DEP forces should collect at those areas.

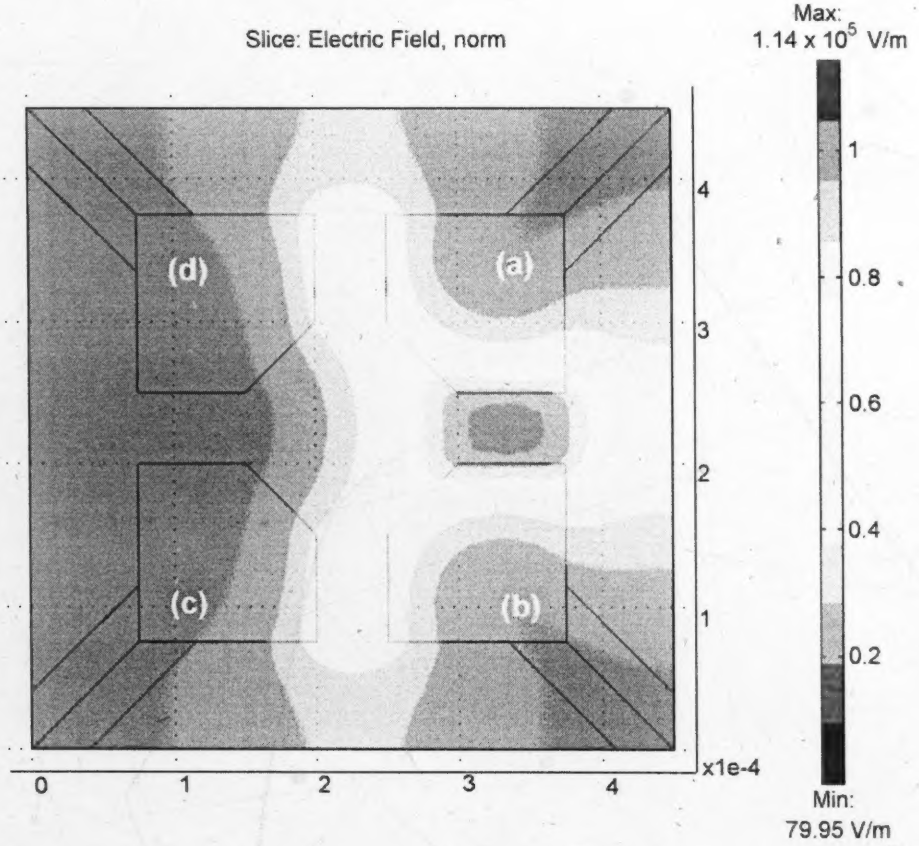


Figure 20. Electric field strength distribution in the x-y plane at z = 25 microns 10 Vpp for the quadrupole electrode structure, electrodes c) and d) are grounded.

5.4.2 Results for White Sulfate Beads

We tested this electrode structure with white sulfate beads and prepared a solution as described in Section 5.3. The pictures show the reactions of the beads at different frequencies upon the application of 5 Vpp on the electrode of the right side. Figure 21.a shows the initial random distribution of the beads. At 1 kHz, beads experienced positive DEP forces, and began to collect along the electrode edges (although many were still settling) but show a jiggly movement due to AC Electro-Hydrodynamic

(EHD) forces (see section 1.1.6), as shown in Figure 21.b. As the frequency increased to 3.9 kHz as shown in Figure 21.c, the positive DEP force affecting the beads started to decrease and beads formed pearl chains. Above 10 kHz, beads experienced negative DEP and collected either on top of the electrodes or away from the structure as shown the series of Figures 21.d through 21.i. This clearly suggests that the crossover frequency for this type of beads should be found between 3.9 kHz and 10 kHz. Electric characterization (ROT or DEP spectra, see section 2.1.2) has not been obtained for this type of beads or for red CML latex beads, at least not to our knowledge.

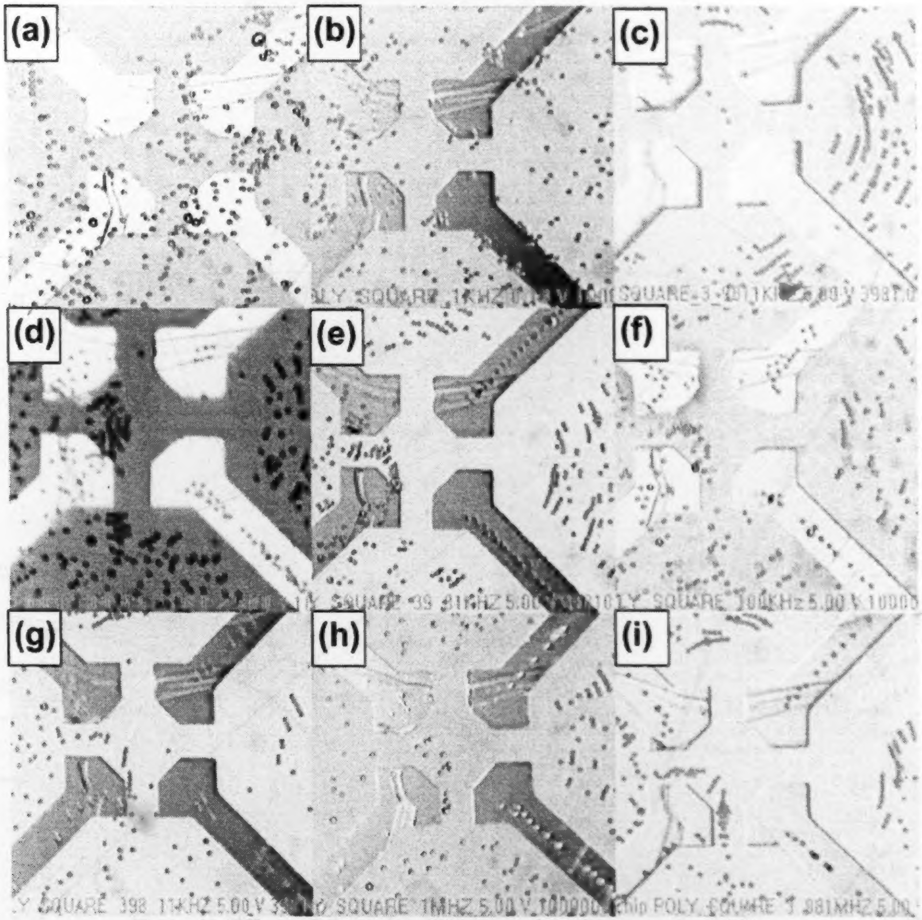


Figure 21. Effects of the electric field on quadrupole structures at different frequencies on white sulfate beads. (a) shows the initial random distribution of white beads. (b) shows the response at the application of 5V at 1 kHz, the frequency was increased to 3.98 kHz in (c), 10 kHz in (d), 39.81 kHz in (e), 100 kHz in (f), 398.11 kHz in (g), 1 MHz in (h), and 3.98 MHz in (i). Beads experienced positive DEP forces up to frequencies of 3.98 kHz. Above frequencies of 10 kHz, beads experienced negative DEP forces.

5.4.3 Results for Surfactant Free Red Carboxylate-Modified Latex Beads

We used the same structure but this time to study the effects on surfactant free red carboxylate modified latex beads (diameter 4 microns). Beads were prepared as described in Appendix B. Five volts peak to peak were applied to electrodes on the right hand; electrodes on the left hand were not polarized (remained floating). Red beads experienced positive DEP up to 15.849 kHz although they oscillated very rapidly at the electrode edges as can be observed from Figures 22.a1 through b. We believe that this oscillatory movement is a consequence of EHD forces.

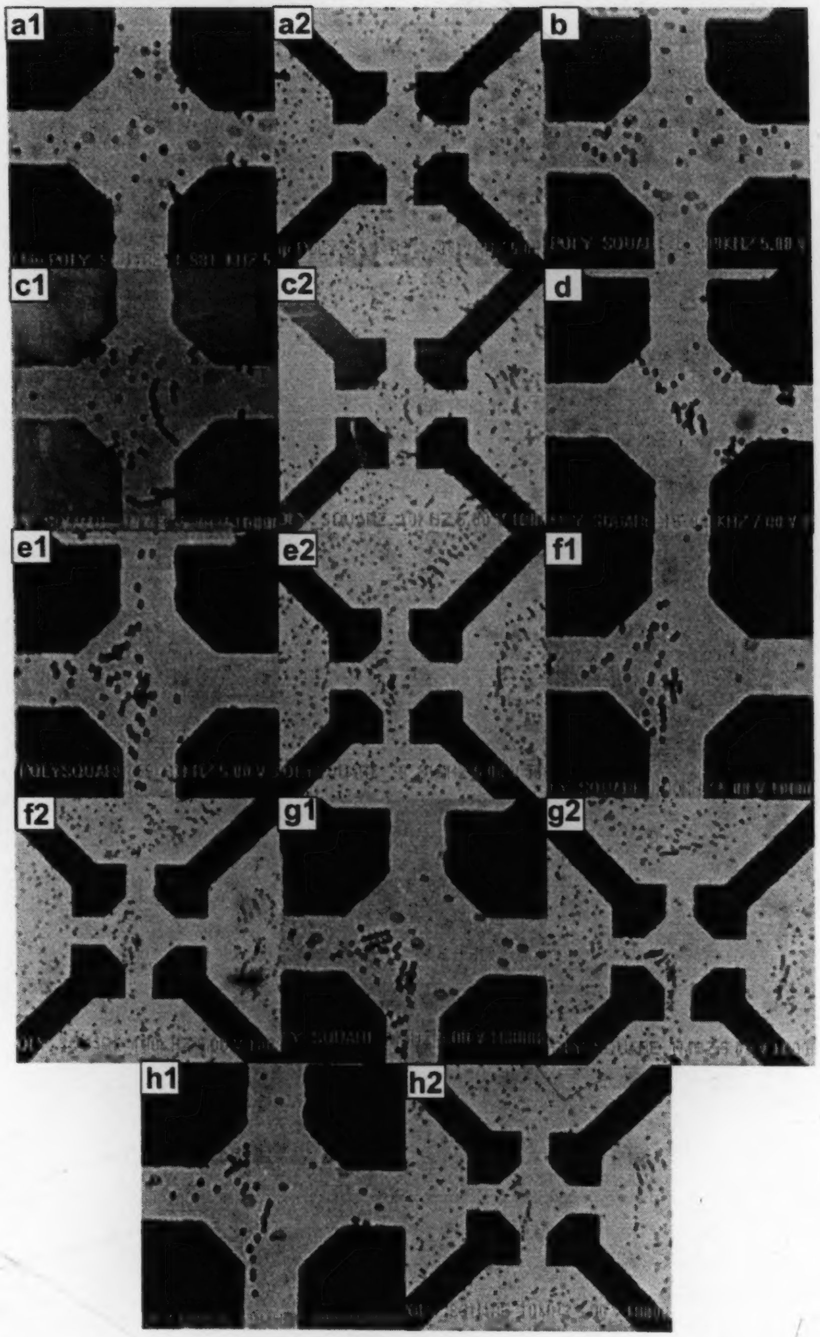

a1
a2
b
c1
c2
d
e1
e2
f1
f2
g1
g2
h1
h2

Figure 22. Effects of the electric field on quadrupole structures at different frequencies on red CML latex beads. The chamber height is 48 microns.

Figure 22.a1 and 22.a2 show the response of red beads at 3.98 kHz. The beads tended to concentrate at the edges of the right side electrodes and oscillated around the edges as a consequence of EHD forces (section 1.1.6); we did not observe any collection or oscillation of beads at the left side electrodes. We turned off the power and let the beads to disperse and settle down every time we switched to a different frequency.

The frequency was increased to 6.309 kHz as shown in figure 22.b and beads still collected at the edges of the electrodes although still affected by EHD forces. We increased the frequency to 10 kHz, and we observed the formation of pearl chains, as can be seen from pictures 22.c1 and 22.c2. At 15.8 kHz, positive DEP forces were still present and beads collected along the edges of the electrodes as shown in Figure 22.d.

Above 39.8 kHz red beads moved to regions were the electric field was minimum as a consequence of negative DEP forces. Shown in Figure 22.e1 and 22.e2, beads are repelled from the electrodes or collect on top of them at 39.8 kHz. We did not observe any significant change in the response of the beads at 100 kHz, 1 MHz and 10 MHz as shown in Figures 22.f1-f2, 22.g1-g2, 22.h1-h2, respectively. That is, the force remained almost constant at that interval of frequencies.

We can then surmise that the crossover frequency for this particle would be found between 15 kHz and 39 kHz; in contrast with white beads, where we found that the crossover frequency should be somewhere below 10 kHz.

5.5 Polynomial Electrode Structure

5.5.1 Simulation

The simulation of the electric field strength distribution for the polynomial electrode structure was performed as described before and is shown in Figure 23. The center area amid the electrodes has a diameter of 64 microns whereas the gap between electrodes is 25 microns. In order to generate the electric field distribution, electrodes to the left and right hand were set at 5 VDC and -5VDC respectively, while electrodes at the top and bottom part were set at 5 VDC and -5VDC respectively. The simulation shown in Figure 23 suggests that high electric field intensity areas are found between electrode gaps, but very weak at the center where the four electrodes converge and on top of the electrodes surface as expected.

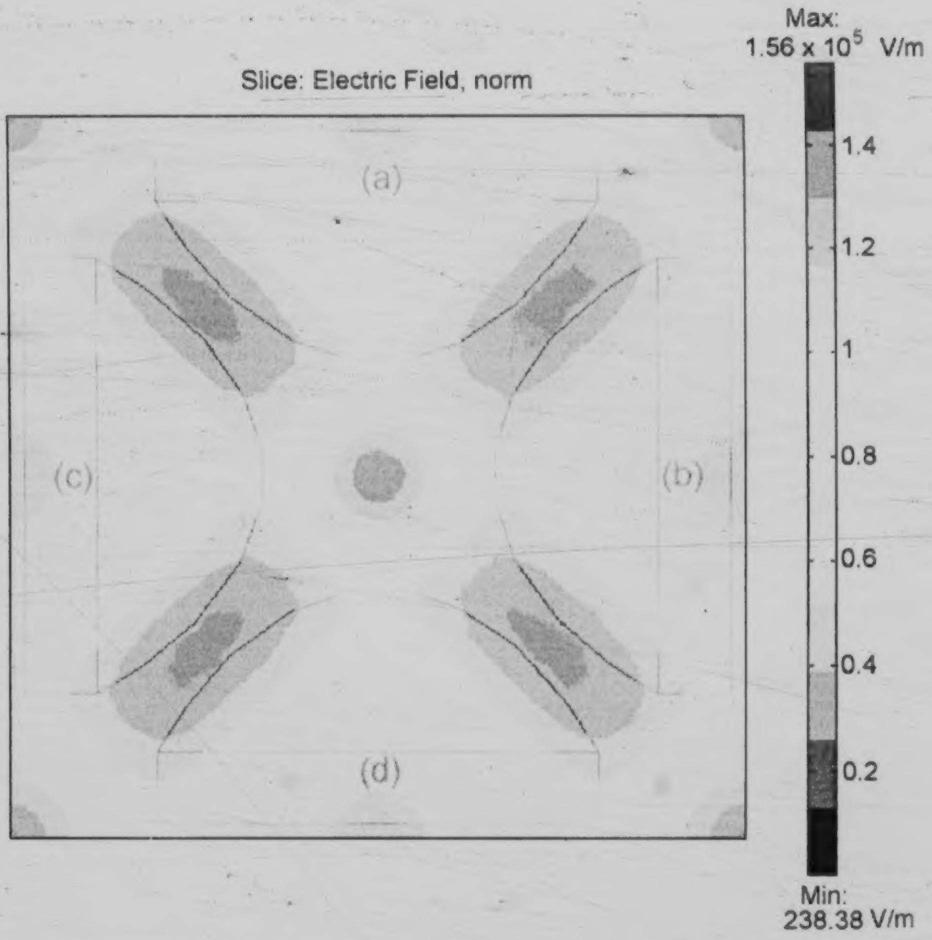


Figure 23. Electric field strength distribution in the x-y plane at z = 25 microns 10 Vpp for the polynomial electrode structure. Electrodes are energized at a = 5V DC, b = -5V DC, c = 5V DC, d = -5V DC.

5.5.2 Results for Yeast Cells

Active dry yeast (Fleischmann's, ON, CA) was harvested from plain broth (LB Broth, Fisher Scientific, NJ, USA) by centrifugation, washed three times in 280 mM Mannitol (D-Mannitol, Fisher Scientific, NJ, USA) and finally suspended in 280 mM Mannitol and stored for 1 month at 4°C. The conductivity was measured to be 6 μS cm^{-1} determined using an Orion 150A conductivity meter.

Figure 24 shows the design of the microfluidic chip used in this experiment. PDMS microfluidic channels had a width of 20 microns. Access ports located at the left were sealed. The solution containing yeast cells was injected at the inlet port. A differential pressure was then created by applying vacuum at the outlet port and yeast cells were dragged down to the separation chamber. Shown in Figure 25.a is the random initial distribution of yeast cells after the application of vacuum at the outlet port.

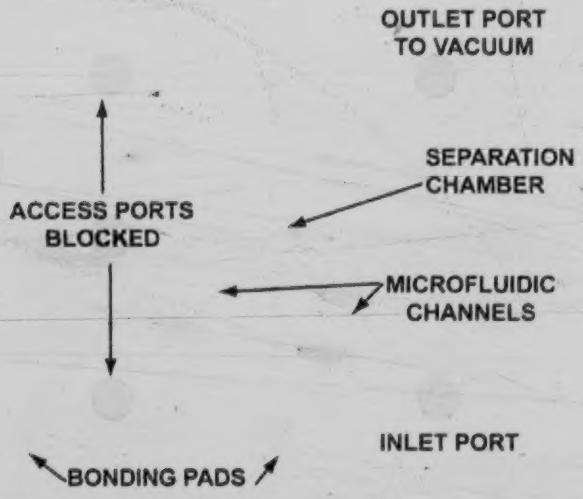


Figure 24. Microfluidic chip containing polynomial electrodes being used to study yeast cells.

Ten volts peak-to-peak were applied to the horizontal electrodes and ground was connected to the vertical electrodes. Upon the application of a voltage at 10 KHz, the cells moved to the edges of the electrodes, as can be observed in Figure 25.b. Every cell that arrived after the electric field was set remained within the boundaries of the electrodes and started to pile up in regions where the electric field was stronger. Observation of negative DEP was also observed when the conductivity of the solution was raised up to 150 μS cm−1, the cells started to collect at the center of the electrodes, however, the electric field was very weak that the force exerted by the liquid flow (due to the difference in pressures) overcame the dielectrophoretic force, nonetheless it could be observed that there was a change in the trajectory of the cells.

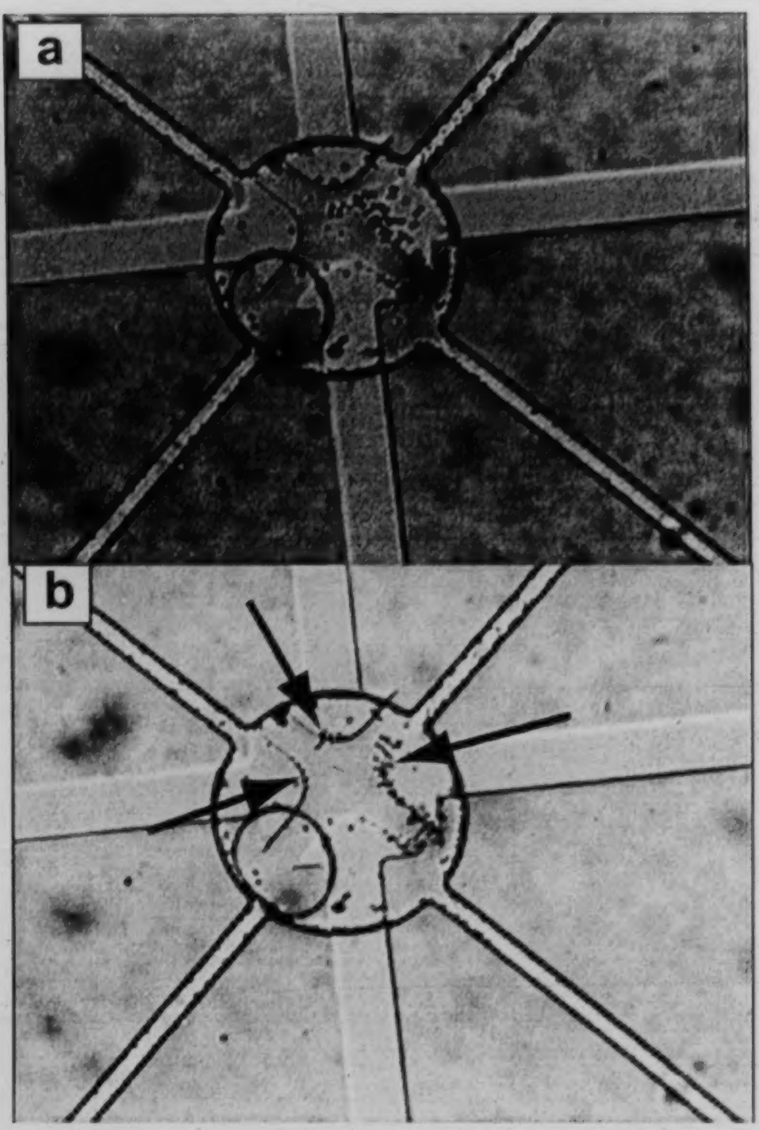


Figure 25. (a) shows the random initial distribution of yeast cells at the polynomial electrode structure, beads are found specially on top of the surface of the right side electrode. At the application of 10 Vpp at 10 kHz yeast cells collected at the edges of the electrode caused by positive DEP forces and formed pearl chains as shown in (b) and pointed by the arrows.

5.6 Interdigitated Electrodes

We studied the interdigitated electrode structure using white sulfate beads and red CML latex beads. The next section shows both simulation and results.

5.6.1 Simulations

The simulation of the electric field strength distribution for the interdigitated electrode structure was performed as described before and is shown in Figure 26. Simulations were performed setting the potential at the electrodes to be +5 V and -5V. The simulation indicates that electric field maxima are found at the edges of the electrodes and in the corners. Electric field minima are found on top of the electrodes and between electrode gaps as shown in Figure 26.

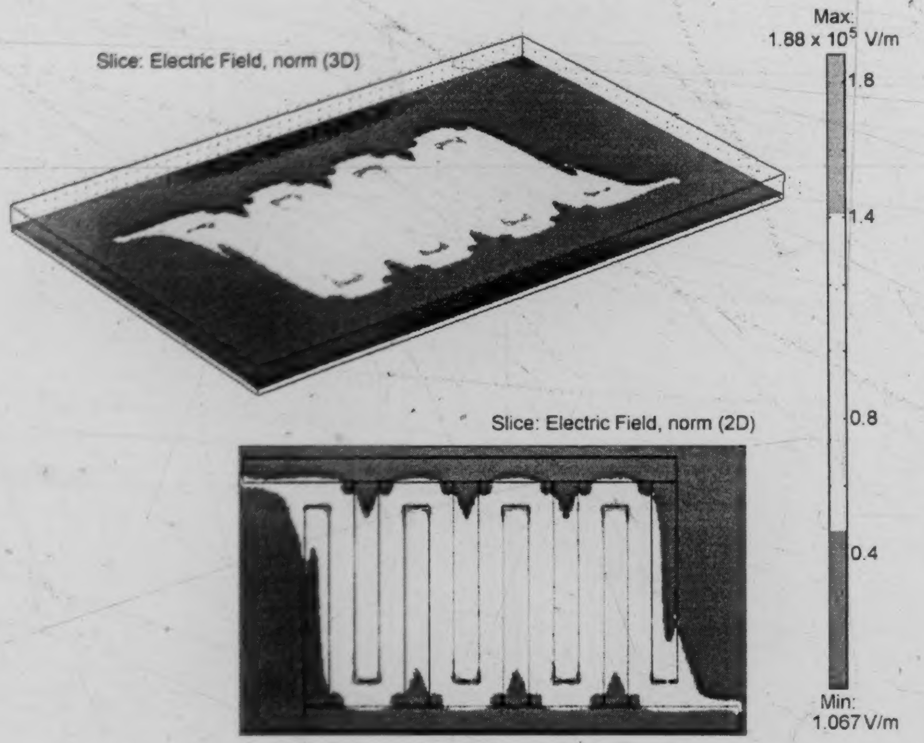


Figure 26. 3D (above) and 2D (below) simulations of the electric field distribution in interdigitated electrodes. High electric field intensities are found at the edges of the electrodes. Weak electric field intensity is found between electrode gaps and on top of the electrode surfaces.

5.6.2 Separation of White and Red Latex Beads

We separated white sulfate beads (9 μm) and red CML beads (4 μm) on interdigitated electrodes with sized and gaps of 20 microns. Beads were prepared as explained before. Figure 27.a shows the initial random distribution of the particles. At the application of 5 Vpp at 5 kHz, the white beads collected on the surface of the electrodes (negative DEP) while red beads gathered at the edges of the electrodes (positive DEP) as shown in Figure 27.b. A higher resolution image is shown in Figure 27.c (white arrows point to red CML beads). Red CML beads have collected at the edges of the electrodes.

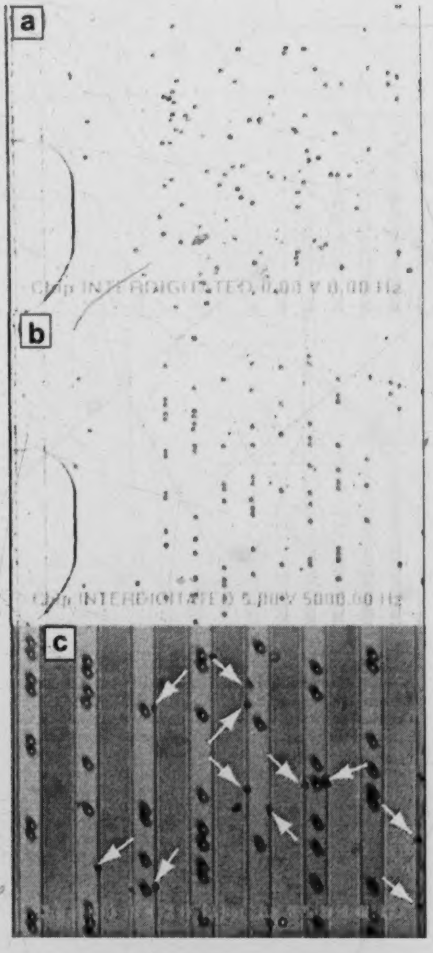

a
Chip INTERDIGITATED 0.00 V 0.00 Hz
b
Chip INTERDIGITATED 5.00 V 5000.00 Hz
c

Figure 27. Separation of red (smaller) and white (larger) beads on interdigitated electrodes. (a) shows the initial random distribution of beads. At the application of 5V pp the red beads collected at the edges of the electrodes while the white beads were trapped on top of the electrode surfaces as shown in (b). Arrows on (c) point to red CML beads.

5.7 Castellated Electrodes

5.7.1 Simulations

The result of the simulations for the distribution of the electric field strength is shown in Figure 28. It shows the distribution in the x-y plane; electrodes were set at +5V, -5V, +5V, and -5 V from top to bottom. Figure 29 shows a simulation in a three dimensional plane and the distribution in the z-x plane to appreciate more clearly that on top of the electrodes the electric field almost vanishes. As it can be observed from the figure, the electric field strength is maximum between the electrode gaps. At the bays of the castells and on top of the electrodes surface the electric field is nearly zero.

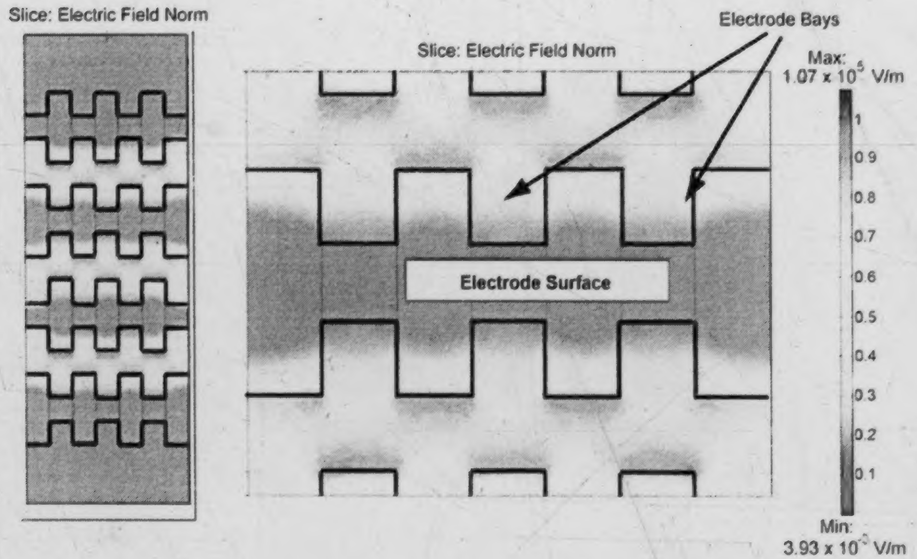


Figure 28. Electric field strength distribution at the castellated electrode (left) and zoom (right) in the x-y plane at z = 25 microns above the plane. Electric field intensity is stronger between electrodes while at the surface of the electrodes and at the electrode bays the electric field intensity is very weak.

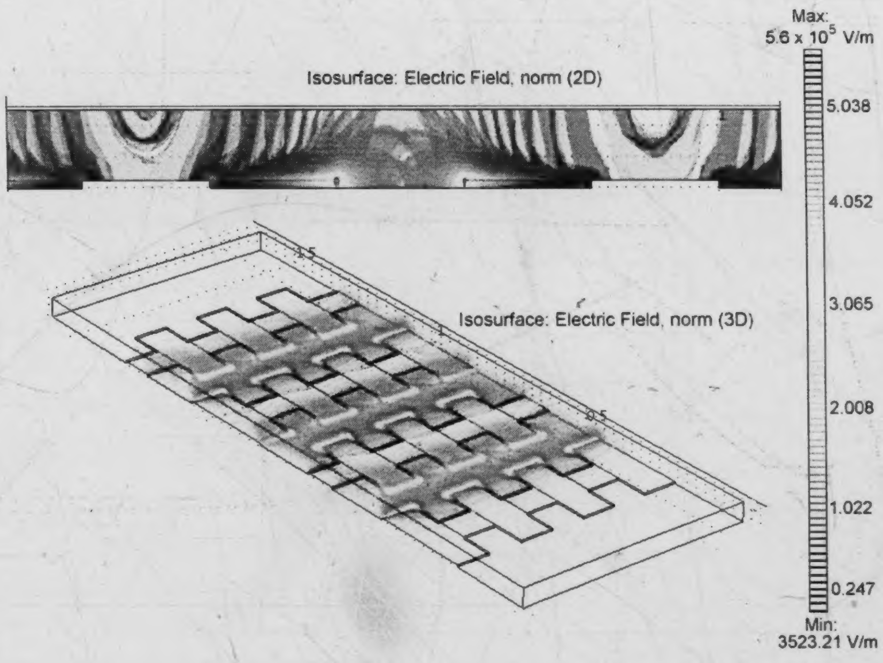


Figure 29. Distribution of the electric field strength in the z-y plane (above); the height of the chamber is 50 microns. Three-dimensional electric field distribution (below) for the interdigitated castellated electrode structure

5.7.2 Results for White Sulfate Beads on Electrodes with Gaps of 80 μm

We studied the effects of the castellated electrodes with gaps of 80 microns on white sulfate beads at different frequencies. Figure 30.a shows the behavior of the particles at the application of 10 Vpp at 1.5 kHz; clearly, beads are collecting between electrode gaps occasioned by positive DEP forces. At a frequency of 5 kHz, beads started to displace to regions where the electric field starts to decrease, that is between castell bays, indicating a decrease in the magnitude of the positive DEP force as shown in Figure 30.b. At 10 kHz cells are collected in the bays of the castell as demonstrated in Figure 30.c. The magnitude of this force started to show variations as the frequency increased. We observed that at 20 kHz single beads were moved to the electrode surface as a consequence of negative DEP. Beads that were close to each other grouped into pearl chains and moved to the castell bays as shown in Figure 30.d. At frequencies of 500 kHz and 1 MHz, all beads moved on top of the electrodes as shown in Figures 30.e and f. These results are in agreement with the previous experiments performed in quadrupole electrode structures with white latex beads because they have the same behaviour for the range of applied frequencies.

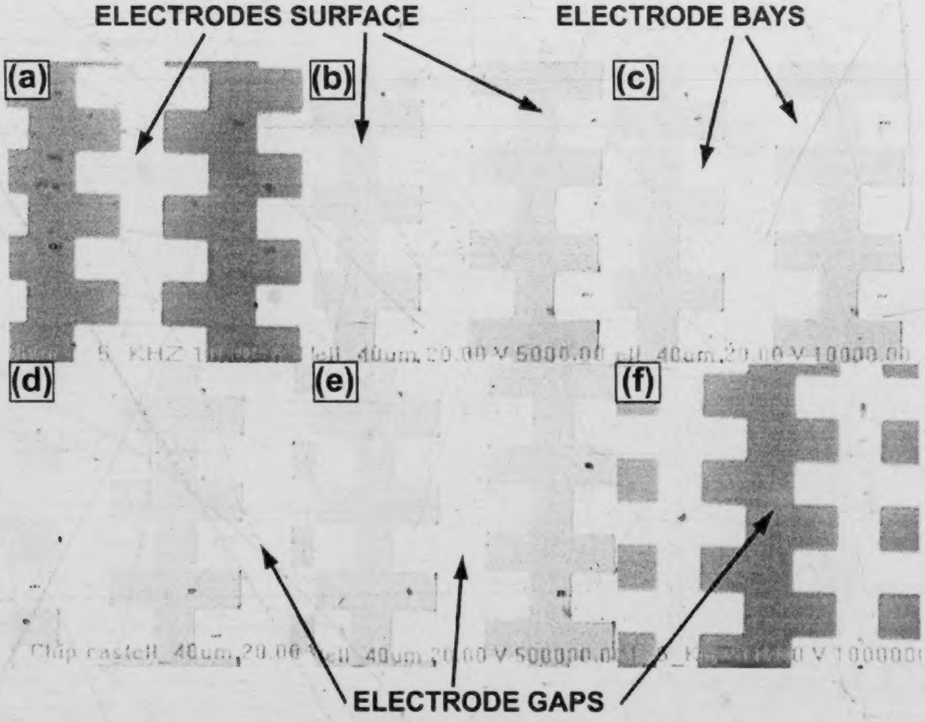


Figure 30. Effects of the electric field on castellated structures at different frequencies on white sulfate beads. Electrode gaps are 80 μm. White latex beads experienced positive DEP up to 10 kHz as shown in Figures a-c. At 20 kHz lonely beads collected on top of the electrodes and pearl chain formations collected at the electrode bays where the electric field is minima as shown in Figures d. Above 500 kHz all beads collected at the surface of the electrodes as a consequence of negative DEP.

5.7.3 Results for Red CML Latex Beads on Electrodes with Gaps of 60 μm

We performed the same experiments for red latex beads, but this time with electrode structures having gaps of 60 microns. We applied 10 Vpp at 5 kHz (Figures 31.a) and 10 kHz (Figures 31.b) and observed that positive DEP forces trapped beads very weakly at the electrode bays. But at 20 kHz and 200 kHz all beads gathered on top of the electrodes as shown in Figures 31.c and 31.d. These results are in accordance with the experiments performed in quadrupole structures for red latex beads (*i.e.* they have the same behaviour for the range of applied frequencies).

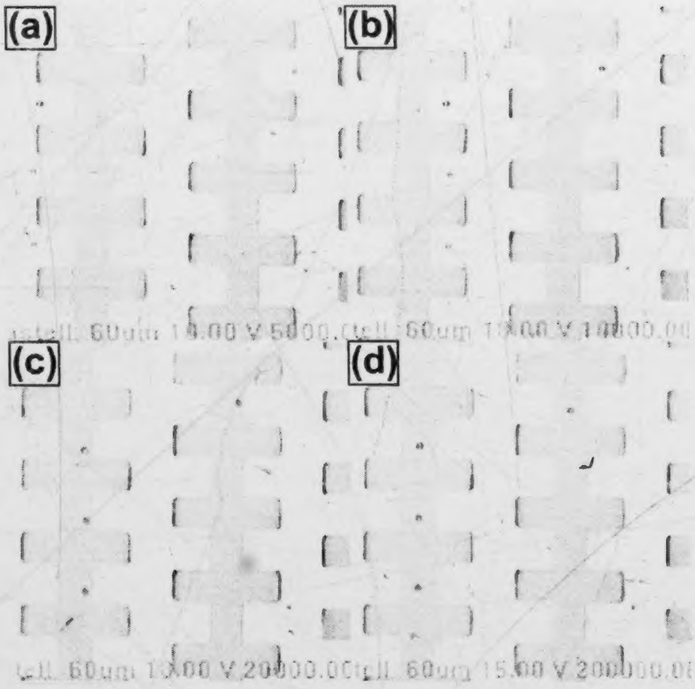


Figure 31. Effects of the electric field on castellated structures at different frequencies on red CML latex beads. Electrode gaps are 60 μm. Beads experiencing positive DEP were pulled very weakly at the electrode gaps upon the application of 10 Vpp at 5 kHz as shown in Figure a. At the application of 10 kHz, the positive DEP force decreased and the beads moved to the electrode bays. Above 20 kHz beads experienced negative DEP and collected on top of the electrodes as shown in (c) and (d).

5.7.4 Results II for Red CML Latex Beads

A solution containing red CML latex beads was injected into a microfluidic chip containing a castellated electrode structure with gaps of 60 microns. The solution was injected at the inlet using a syringe and it was left in the chamber for a few seconds until no bead movement was observed (as shown in Figure 32.a). Then, upon the application of 10 Vpp at 5 KHz, red beads started to accumulate at the edges of the electrodes where the electric field is stronger and aggregate in the form of pearl chains as shown in Figure 32.b. They were retained by positive DEP forces. One second later, Figure 32.b, most of the beads were trapped, except for some of them that were oscillating near the edges and on top of the electrodes, perhaps caused by Joule heating as shown by Figures 32.c, d, and f. Figure 32.g shows how more particles tend to gather at the bus bars of the electrodes where the electric field is stronger than anywhere else in the structure.

Then, we created a pressure difference between the inlet and the outlet port by adding more solution to the inlet port so fluid could start moving again. As shown in the snapshots 32.h, i, and j, red beads were trapped at the very first electrode bars, until that area became saturated as shown in Figure 32.k.

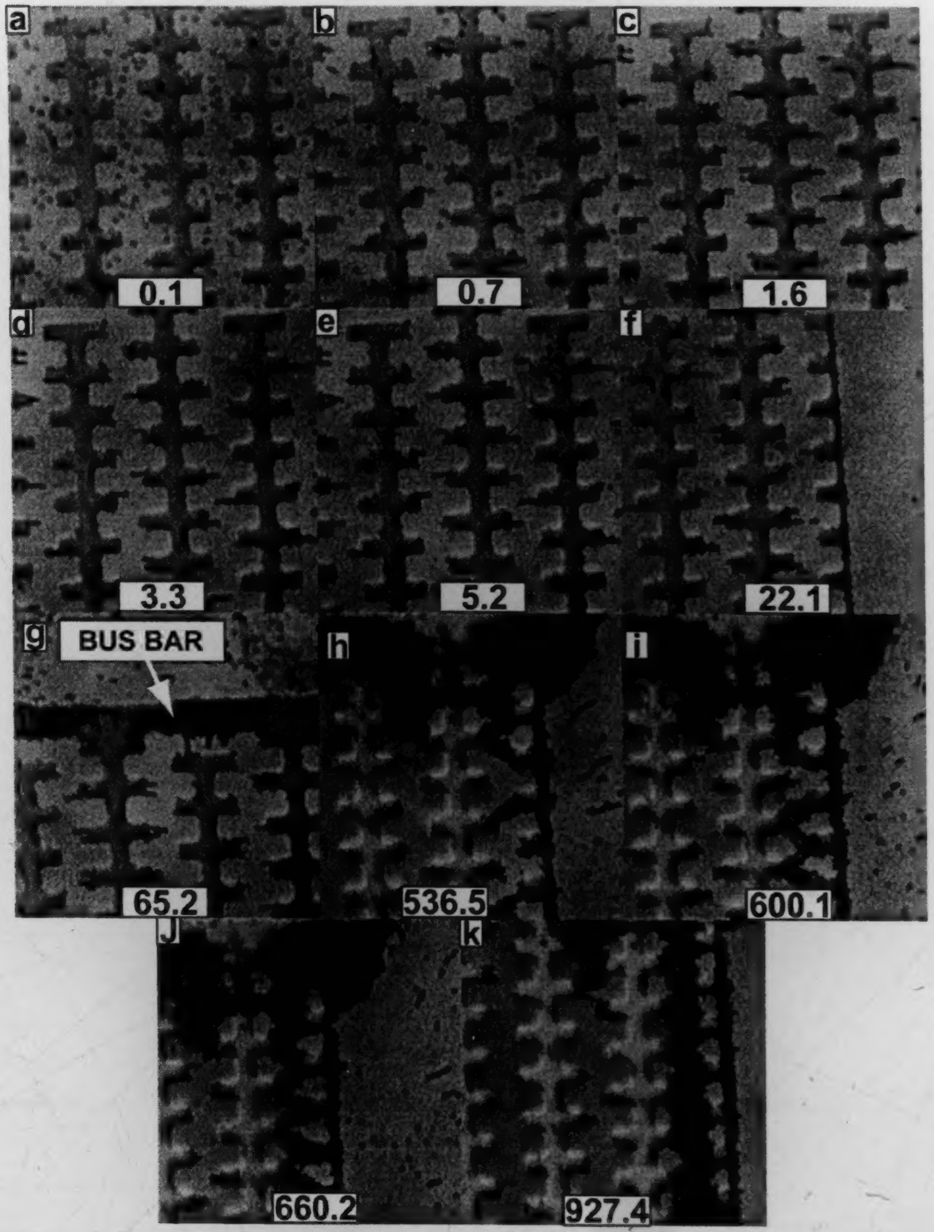

a
b
c
0.1
0.7
1.6
d
e
f
3.3
5.2
22.1
g
BUS BAR
h
i
65.2
536.5
600.1
J
k
660.2
927.4

Figure 32. Effects of the electric field on castellated structures at a fixed frequency on red CML latex beads. Electrode gaps are 60 μm.

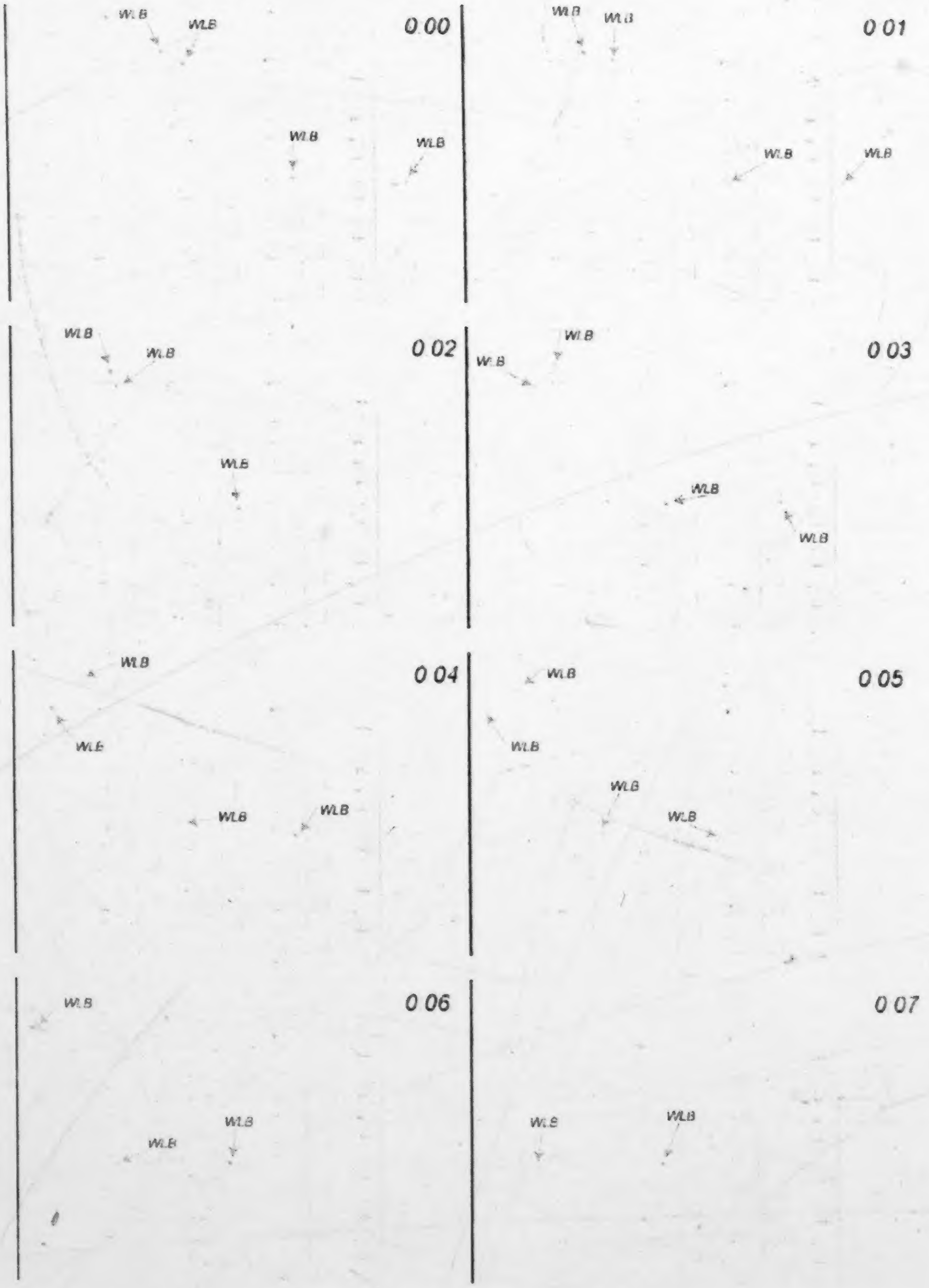
5.7.5 Separation of White and Red Latex Beads

Castellated electrodes with gaps and dimensions of 80 microns were employed for this experiment. A solution containing sulfate white beads and CML read beads coated with BSA and diluted in deionized water was measured to have a conductivity of 3 μS/cm and pH of 8.4. The solution was loaded into one port of the channel. Then vacuum was connected for one second to the other port to initiate the flow. Electrodes were energized at 10 Vpp and 2500 Hz 30 seconds after the solution was loaded.

Figure 33 shows a series of frames demonstrating the separation of white beads from red beads. Red beads remained trapped at the electrodes because of positive DEP forces whereas the white beads experienced negative DEP. The force of the fluid drag was enough to transport the white beads along the channels. This force was greater than the negative DEP force experienced by the white beads and thus white beads moved with the stream flow. However, it was not enough strong to release the red beads already collected at the edges of the electrodes.

The red arrows are pointing to white latex beads on the series of images in Figure 33. In the first frame, a population of red latex beads has been already trapped in the electrodes by positive DEP forces. Also shown, in the same frame, are four white beads which are experiencing negative DEP. A pressure difference was created by increasing the amount of liquid at one of the ports. In this way, fluid flow started to drag white beads to the left side.

As it can be seen in the next 10 frames (elapsed time: 10 seconds), none of the white beads have been trapped. The reason is that the crossover frequency for the white beads is ~ 3 KHz, whereas for the red beads is ~ 40 KHz, so the force acting on the white beads was almost null and red bead experienced a positive dielectrophoretic force. The remaining four snapshots show none white beads captured and only red beads accumulating at the edge of the electrodes due to positive dielectrophoretic forces. Applying vacuum to one of the ports flushed away all the beads. We could repeat the experiments as long as the pressure difference between the two ports was not too high. We created this pressure difference by adding more solution to one of the wells, although we did not have any way to quantify or measure this difference. Future work should address this issue.


0 00
WLB
WLB
WLB
WLB
0 01
WLB
WLB
WLB
WLB
0 02
WLB
WLB
WLB
0 03
WLB
WLB
WLB
WLB
0 04
WLB
WLB
WLB
WLB
0 05
WLB
WLB
WLB
WLB
0 06
WLB
WLB
WLB
0 07
WLB
WLB

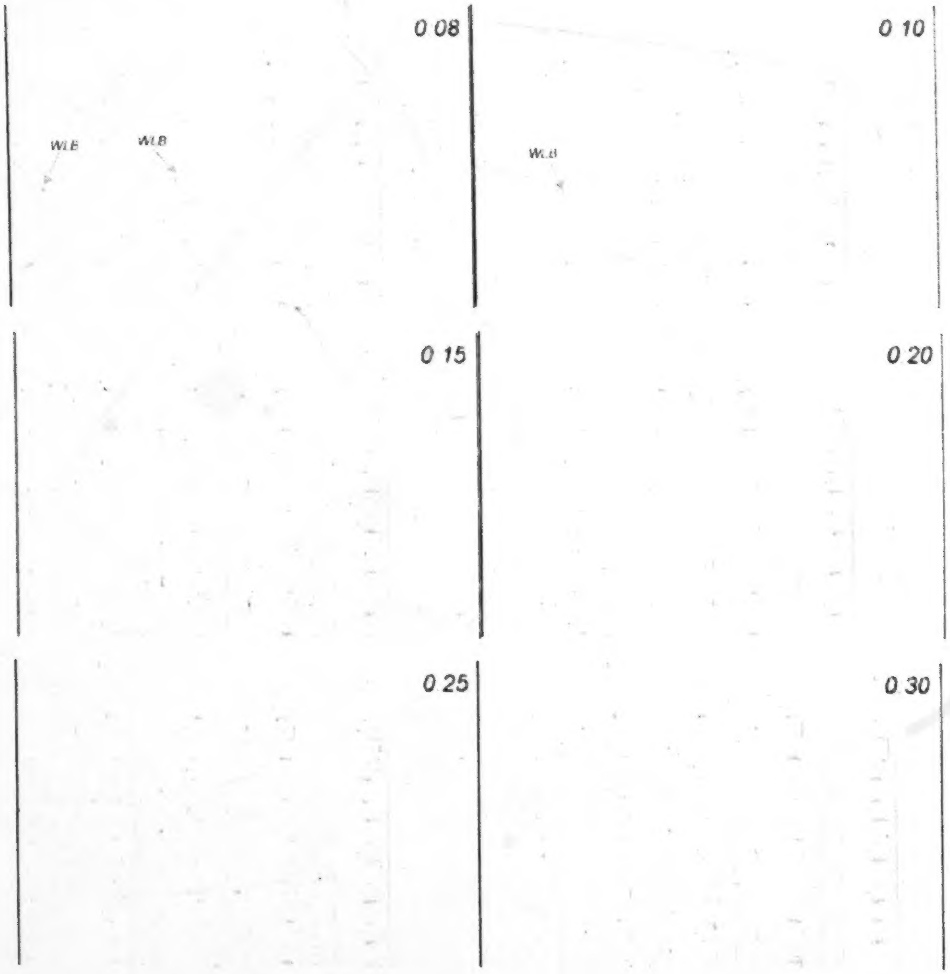


Figure 33. Separation of red and white beads on castellated electrodes with gaps of 80 μm. Time elapsed is shown in seconds in the upper right corner on every frame. WLB indicates white latex beads. At time zero red beads are trapped at the electrode edges by positive DEP while white beads experiencing negative DEP are transiting through the electrode. The stream flow started to drag white beads along the electrodes until they were eluted (1 second to 10 seconds), because the drag force was enough to overcome the negative DEP force experienced by white beads. The last four frames show that only red beads remained trapped at the electrodes.

Chapter 6

6 HARDWARE AND SOFTWARE TOOLS

6.1 Function Signal Generator

We realized that the advantages provided by DEP-based microfluidic chips, especially cost and space, were negated by some of the equipment necessary to operate them. In our case, electrical signals powering the chip were supplied through a function signal generator as explained in the previous chapter. This signal generator costs ~ US \$2000 and has a width of 25.44 cm, a length of 37.4 cm and a height of 8.85 cm. Our microfluidic chip has a width of 2.5 cm, a length of 5 cm, and a height of ~0.5 cm and cost ~\$US 20. This disparity in costs and dimensions jeopardizes the use of DEP-based microfluidic chips. We therefore built an inexpensive (cost ~US\$140) and compact function signal generator.

There are three possible ways to generate waveforms at specified amplitudes and frequencies to power DEP-based chips. Each of them presents its own advantages and drawbacks in their functionality and when trying to integrate them with a DEP-microfluidic chip. A desirable device should be one that is portable, easy-to-implement and control, compact and small, multifunctional, operates in a wide range of frequencies and voltages, able to output at least 4-phases with arbitrary waveform functions, and inexpensive (for foreseeable mass-production).

The first one is to design an electronic microchip based on CMOS technology or BJT. Different modules for the frequency generator such a program and control modules and amplification stages would be required. However, different electronic chips exist in the market nowadays that provides these functionalities such as Direct Digital Synthesizers.

The second one, and maybe the easiest way, is to employ an instrument "off-the-shelf"; like the Agilent 33120A (CA, USA) we used in the previous chapter. Among its many characteristics are the simplicity to program it from any computer through the serial port (RS-232) or a GPIB interface (IEEE-488) which translates in time-saving implementations, allows the creation of arbitrary waveforms, provides an output voltage range of 10 Vpp (50Ω) and 20 Vpp (open circuit), and provides a maximum frequency of 15 MHz for a sine or square wave function. However, its dimensions make it impractical for any future integration or commercial application with a μ-fluidic chip. Furthermore, its cost (estimated price US \$2097) counteracts the assets described above. In any case, the presence of this instrument is desirable during the first stages of the experiments.

The last one and simplest alternative is to have a PCB with the appropriate chips to generate the waveform functions. To this end, we designed an electric circuit capable to supply the necessary signals for the electrodes, signals with maximum amplitude of 12 V and maximum frequency of 10 MHz.

The architecture of this frequency generator revolves around a 50 MHz CMOS Complete Direct Digital Synthesis (DDS) chip, AD9835, manufactured by Analog Devices (MA, USA). A digital potentiometer AD8400 (Analog Devices, MA, USA) was also included to vary the amplitude of the output signals.

According to the manufacturer, the DDS chip can generate sinusoidal voltages with frequencies up to 25 MHz, and the phase of the output signal can be preprogrammed, which for our use is advantageous since at the moment there are no instruments on the market which can supply two output signals 90° phase shifted in the desired range of frequencies. Furthermore, it is possible to have a complete control of the output frequency and phase in real time from a computer. Even when there are many chips in the market with similar characteristics or possible configurations, this design is to be believed more robust than others because of the direct control exerted on the frequency and phase of the signal as well as the amplitude.

A brief description of the circuit is given below. A detailed list of all components can be found in appendix B. The Printed Circuit Board (PCB) layout is included in Appendix D. Circuits and PCB layouts were created in the Easily Applicable Graphical Layout Editor (Eagle v.11 for windows, CadSoft Computer, Inc., FL, USA).

The diagram shown in Figure 34 gives a general overview of the design. The complete schematic can be found in the appendix. Our custom board includes a DDS, a digital potentiometer, connectors for the power supplies (+5V, +15V, -15V, GND) and for the microcontroller, a 50 MHz clock, an amplification stage, and connectors for the outputs. The DDS and the digital POT are programmed via the Inter-IC (I^2C) serial bus (consists of a two-wire bus, serial data SDATA and serial clock SCLK). I^2C signals are generated on a custom board containing a microcontroller (PIC16F877, Microchip Technology Inc., AZ, USA). The microcontroller board is controlled from the serial port (RS-232) of a computer. Software programs for the microcontroller are included in appendix E.

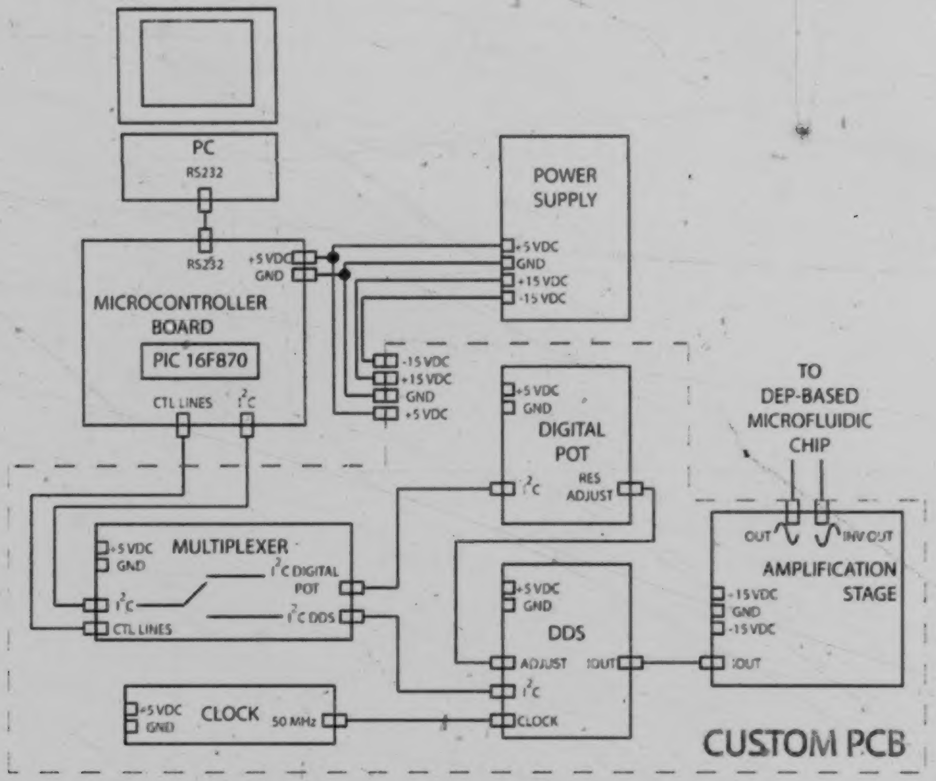

PC
RS232
RS232
MICROCONTROLLER BOARD
PIC 16F870
+5 VDC
GND
CTL LINES
I²C
POWER SUPPLY
+5 VDC
GND
+15 VDC
-15 VDC
-15 VDC
+15 VDC
GND
+5 VDC
DIGITAL POT
+5 VDC
GND
I²C
RES ADJUST
TO DEP-BASED MICROFLUIDIC CHIP
OUT
INV OUT
AMPLIFICATION STAGE
-15 VDC
GND
-15 VDC
IOUT
MULTIPLEXER
+5 VDC
GND
I²C
CTL LINES
I²C DIGITAL POT
I²C DDS
DDS
+5 VDC
GND
ADJUST
IOUT
I²C
CLOCK
CLOCK
+5 VDC
GND
50 MHz
CUSTOM PCB

Figure 34. Diagram shows the main components of the custom function signal generator. The microcontroller can be programmed from a PC through the serial port. The digital potentiometer and the direct digital synthesizer are controlled via the I²C protocol. I²C signals generated in the microcontroller are multiplexed to the digital potentiometer and to the direct digital synthesizer. Dashed lines encompass the components in the custom PCB.

The DDS and the digital potentiometer have a serial interface (similar to the I^2C); however, it is not possible to address them individually. Thus, we decided to multiplex the signals (as shown in Figure 35) using a Dual 4- channel analog multiplexer-demultiplexer (CD4052BCM, Fairchild Semiconductors, ME, USA). We used the set of I^2C signals generated by the microcontroller to control the DDS and the digital potentiometer. The schematic also shows the connectors to interface the microcontroller and for the power supplies. Control lines are needed to control the select the channels in the multiplexer.

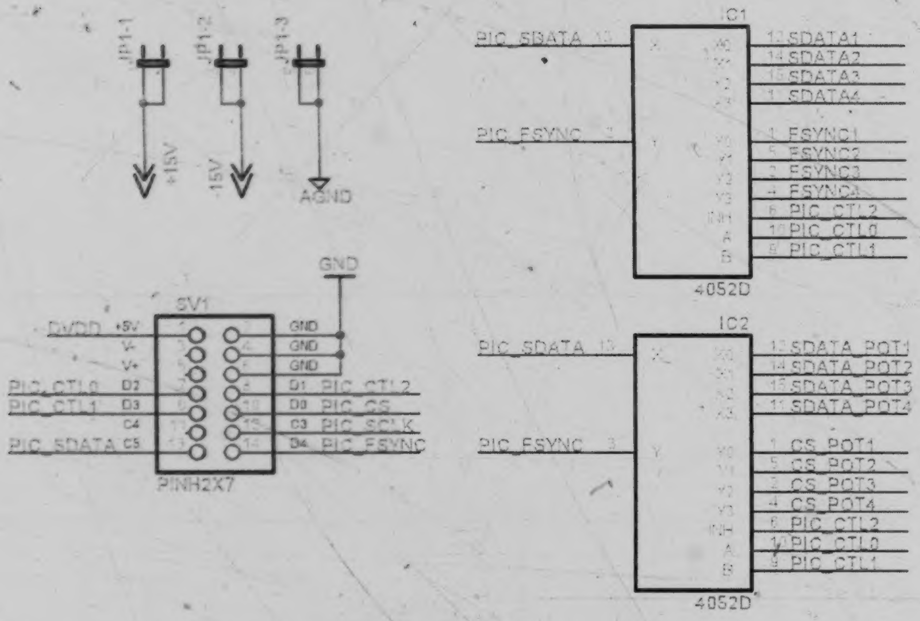

JP1-1
JP1-2
JP1-3
+15V
-15V
AGND
GND
SV1
DVDD
+5V
V-
V+
PIC_CTL0
D2
PIC_CTL1
D3
C4
PIC_SDATA
C5
GND
GND
GND
D1
PIC_CTL2
D0
PIC_CS
C3
PIC_SCLK
D4
PIC_FSYNC
PINH2X7
IC1
PIC_SDATA
PIC_FSYNC
X
Y
X0
X1
X2
X3
Y0
Y1
Y2
Y3
INH
A
B
SDATA1
SDATA2
SDATA3
SDATA4
FSYNC1
FSYNC2
FSYNC3
FSYNC4
PIC_CTL2
PIC_CTL0
PIC_CTL1
4052D
IC2
PIC_SDATA
PIC_FSYNC
SDATA_POT1
SDATA_POT2
SDATA_POT3
SDATA_POT4
CS_POT1
CS_POT2
CS_POT3
CS_POT4
PIC_CTL2
PIC_CTL0
PIC_CTL1
4052D

Figure 35. Schematic shows the connectors for the power supplies and to interface the microcontroller. I^{2}C signals are multiplexed to control the DDS and the digital pot.

The frequency stage is composed of the DDS chip, AD9835, and a digital potentiometer, AD8400, as shown in Figure 36. This design is based in an application note [115]. A 50 MHz fixed frequency SMD oscillator (CSX750FCC50.000MTR, Citizen, CA, USA) supplies the clock signal to the DDS. The digital potentiometer controls the amplitude of the output signal in a range of 256 steps. The early introduction in the control of the amplitude accounts for less noise than it would be if it was introduced in the amplification stage. A potentiometer in the amplification stage would degrade the signal due to parasitic capacitances.

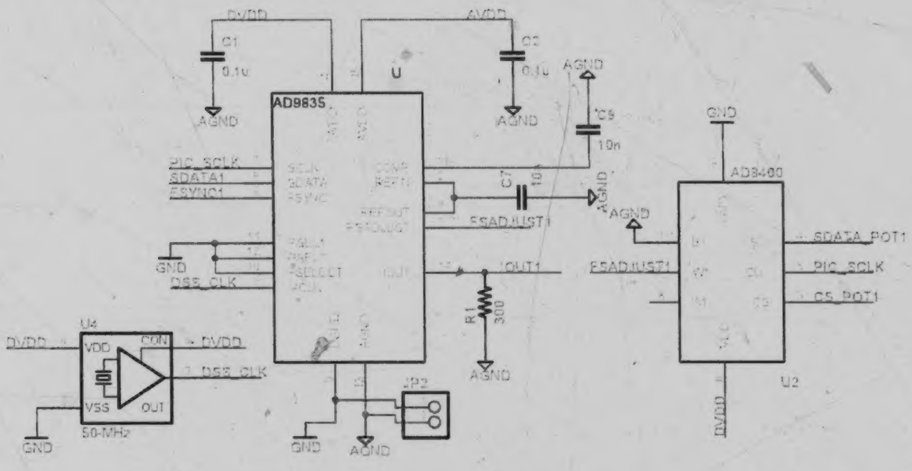

DVDD
C1
0.1u
AGND
AVDD
C2
0.1u
AGND
U
AD9835
AGND
C9
10n
C7
10
AGND
PIC_SCLK
SDATA1
FSYNC1
FSADJUST1
GND
DSS_CLK
IOUT1
R1
300
AGND
GND
AD8400
AGND
SDATA_POT1
FSADJUST1
PIC_SCLK
CS_POT1
U2
DVDD
U4
DVDD
VDD
CON
DVDD
DSS_CLK
VSS
OUT
50-MHz
GND
JP2
GND
AGND

Figure 36. Schematic comprising the frequency generator stage. The schematic shows the wiring of the DDS, Digital POT and the 50 MHz clock.

The DDS provides a maximum current of 4 mA (1.35 V max) at its output pin, so the signal had to be amplified to reach our desired specifications. The amplification stage comprises a current to voltage (I-to-V) converter op-amp configuration, DC shifter, and a fixed-gain amplification.

The I-to-V converter amplifies the signal by a factor of eleven using non-inverting configuration op-amp with a high performance video op-amp (AD811, Analog Devices, MA, USA). The signal coming out of the I-V converter is positive. To make use of the maximum voltage output swing provided by most generic amplifiers, it was decided to extract the DC (RMS) signal using an array of capacitors (rectifier 100 μF, 10 μF and 1 μF). This signal was then subtracted from the signal coming out of the I-to-V converter signal using a summing amplifier configuration; as a result, the DC value of the signal was shifted down by the RMS voltage value. Figure 37 shows the schematic.

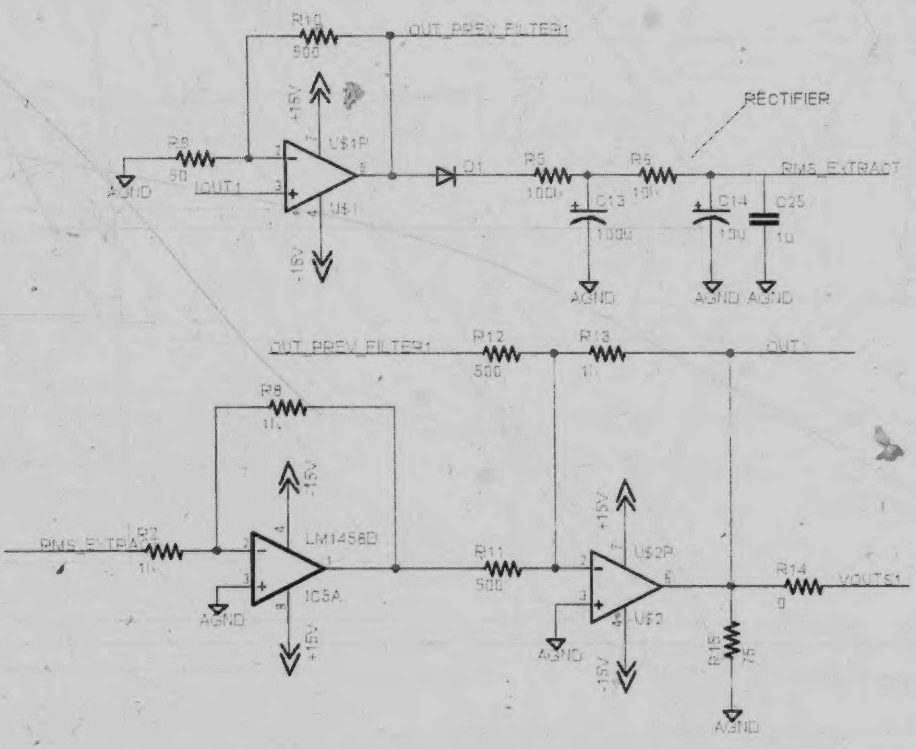

R10
500
OUT_PREV_FILTER1
+15V
RECTIFIER
R9
50
AGND
IOUT1
U$1P
U$1
-15V
D1
R3
100k
R5
10k
RMS_EXTRACT
C13
100u
C14
10u
C25
1u
AGND
AGND
AGND
OUT_PREV_FILTER1
R12
500
R13
1k
OUT
R8
1k
-15V
LM1458D
RMS_EXTRACT
R7
1k
IC5A
AGND
+15V
R11
500
+15V
U$2P
U$2
AGND
-15V
R14
0
VOUT51
R15
75
AGND

Figure 37. Circuit showing the current-to-voltage op-amp configuration and the DC level shifter. See text for details.

The fixed gain amplification consists of an inverter op-amp configuration followed by a unity gain buffer as shown in Figure 38. The inverter shifts the signal coming from the circuit described above 180° apart. Both signals, non-inverted and inverted, are connected to a unity gain buffer terminated in 75Ω resistors. A ribbon connector (53261-0490, Molex-Waldom Electronics Corp., IL, USA) is used to output the signals. This configuration provides maximum output swings of 12V.

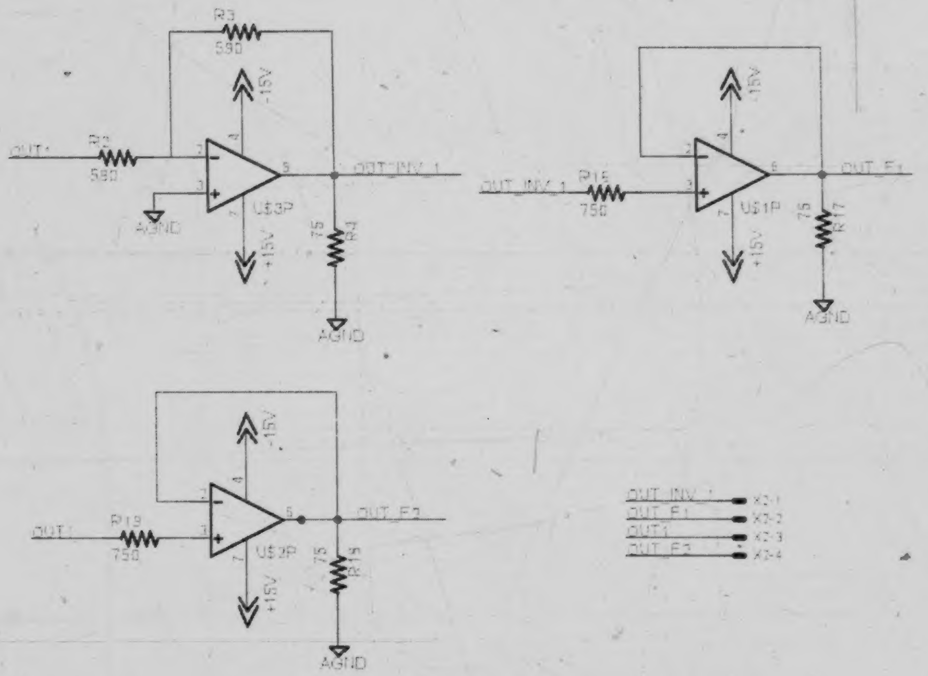

R3
590
-15V
OUT1
R2
590
2
3
4
5
7
U$3P
AGND
+15V
OUT_INV_1
75
R4
AGND
-15V
OUT_INV_1
R15
750
2
3
4
6
7
U$1P
+15V
OUT_F1
75
R17
AGND
-15V
OUT1
R19
750
2
3
4
5
7
U$2P
+15V
OUT_F2
75
R19
AGND
OUT_INV_1
OUT_F1
OUT1
OUT_F2
X2-1
X2-2
X2-3
X2-4

Figure 38. Fixed gain amplification stage. Signal OUT1 is inverted using an inverter op-amp. Signals OUT1 and OUT_INV1 go then to unity gain buffers terminated on 75 ohm resistors.

6.1.1 Results

Output signals were measured with a two-channel color digital phosphor oscilloscope (TDS3032, Tektronix, OR, USA). Collected data was transferred to a PC and viewed with WaveStar ver 2.7.4 (Tektronix, OR, USA). Figure 39 shows 4 readings at different frequencies. Figure 39.a shows the output sinusoidal signal at 1 MHz 12 Vpp. Figure 39.b shows the response at 5 MHz 10.4 Vpp. Figure 39.c presents a signal at 10 MHz 10.3 Vpp. Finally, Figure 39.d shows the response at 20 MHz 10.8 Vpp.

As it can be seen from the figures, signals do not show any sign of considerable degradation up to 10 MHz. Although, the manufacturer states that it is possible to generate signals up to 25 MHz, the measured signal at 20 MHz (Figure 39.d) was greatly deteriorated probably as a consequence of noise in the traces or due to the output signal of the clock. However, our objective to generate sinusoidal signals below 10 MHz was accomplished.

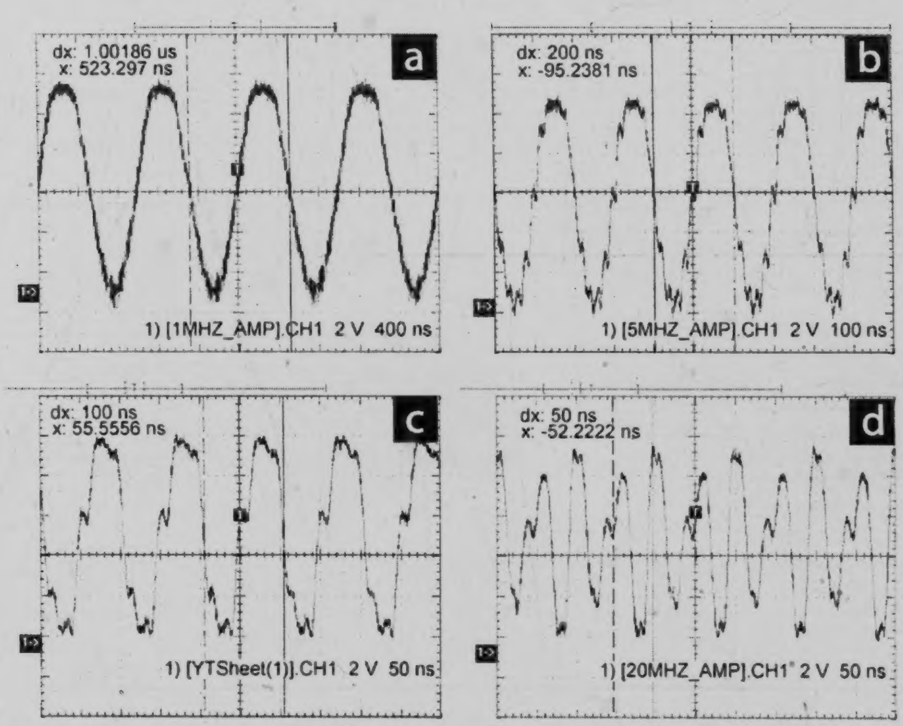


Figure 39. Output signals from the custom signal function generator were measured with an oscilloscope. (a) shows the signal at 1 MHz 12 Vpp, (b) at 5 MHz 10.4 Vpp, (c) at 10 MHz 10.3 Vpp, and (d) at 20 MHz 10.8 Vpp.

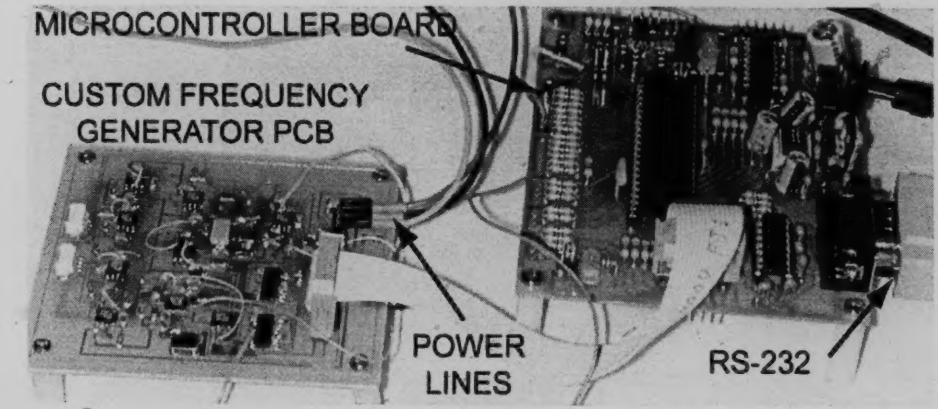

MICROCONTROLLER BOARD
CUSTOM FREQUENCY GENERATOR PCB
POWER LINES
RS-232

Figure 40. Custom frequency generator setup showing the custom PCB and the microcontroller board. Modifications in the circuit are reflected in the schematics previously described, but not in the layouts.

6.2 Software

We decided to create a graphical user interface (GUI) to speed the time taken to perform an experiment and to help manage the large amounts of data produced. This interface displays and records video in real time, adds close captioning to it with relevant information to the experimenter. It takes a snapshot (bmp files) every 100 milliseconds from the video taken and stores the image with a specific name. It also stores information regarding the type of cells, chip design, medium parameters, and comments into a text file.

The interface is shown in Figure 41. The GUI was programmed and compiled in Visual C++ 6 (Microsoft, WA, USA) because it facilitates the creation of windows forms, consoles, and executable programs, and because it is possible to display and record video in real time within the same application.

The program allows to store in a text file the parameters and settings currently used for the experiment, as well as the date. It generates an automatic name for the file based on the chip name, the beads under test, and the local hour, stamped with extension ".dep".

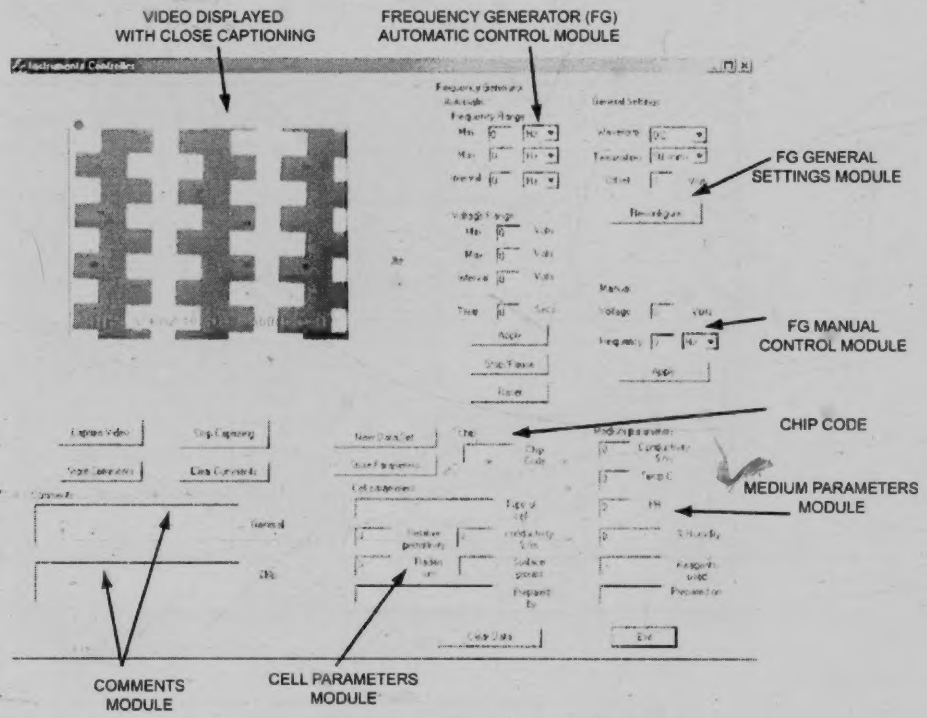

VIDEO DISPLAYED WITH CLOSE CAPTIONING
FREQUENCY GENERATOR (FG) AUTOMATIC CONTROL MODULE
FG GENERAL SETTINGS MODULE
FG MANUAL CONTROL MODULE
CHIP CODE
MEDIUM PARAMETERS MODULE
COMMENTS MODULE
CELL PARAMETERS MODULE

Figure 41. Graphical User Interface used to display video, control the frequency generator, and store comments and parameters relevant to DEP experiments. See text for more details.

As shown in Figure 41, there are different modules. The Cell Parameters Module contains edit boxes to be filled with information relevant to the cell such as: type of cell, radius (μm), conductivity, relative permittivity, surfaces charges, and person who prepared it. The Medium Parameters Module contains 6 edit boxes: conductivity, temperature, and pH of the solution, date of preparation and the type of reagents used. Information about the chip is put in the Chip Code Module. General comments and specific comments about the chip can be written down in the Comments Module. The Capture Module displays the video with close captioning. General comments on the experiment or on the chip can be saved at any time to the file as long as the Clear Data button is not pressed.

Frequency Generator Modules control the function generator (HP33120A, Hewlett-Packard, USA) through the RS-232 in two modes. The Manual Control Module allows setting the frequency and voltage at a time. The Automatic Control Module permits the user to set minimum and maximum limits for the voltages and frequencies, and to set the increment value for sweeping voltage and frequencies in a period of time. This provides a lot of functionality at the time of experimenting because is too time- consuming, especially for rapidly evolving systems, to press buttons on the function generator to increase or decrease the voltage and frequency. The General Setting Module can change the output impedance, as well as the type of waveform function.

6.2.1 Close Captioning

Analysis of data in DEP-experiments is performed visually. This means that images are our source of information. There are many available commercial software programs that offer the possibility to store the video digitally. Nonetheless, these programs do not provide the option to add text or closed caption in real time to the video currently recorded.

The idea of having the video capture within the application was to add close captioning to the video and to display parameters of interest in real time (*e.g.* frequency, voltage, and type of cell) but also to store the video with those parameters. This eventually would translate in simpler future analyses, which in turn would consume less time. A technician could just sit and click in the mouse buttons. Software code can be found in Appendix F.

DirectX 9.0 (Microsoft, WA, USA) is a set of multimedia application programming interfaces (APIS's) developed by Microsoft (compatible with Windows 98, 2000, XP, and NT) and it is distributed for free (http://www.microsoft.com/windows/directx/downloads/default.asp). Obviously, these tools work better with Microsoft Technologies, so we decided to program the application in Microsoft Visual C++ 6.0 based on DirectX. Direct X is a set of components mostly used for the creation of video games and high performance multimedia applications. Each component targets specific applications such as sound and music, graphics, networked applications, and data-multimedia streams.

The "DirectX Show" component supports streaming media on Windows. This was the module that we used to capture and display video and to add close captioning to it.

Our application displays video with close captioning by superimposing a static image (bitmap file containing the information to be displayed: type of chip, frequency and voltage applied) to the captured video as shown in Figure 42. It stores the video to a file in AVI format and acquires and stores automatically 1 frame every 100 millisecond to a bmp file. All files are stamped with the chip name, the type of particles under test, the hour at which the video starts recording, the voltage, the frequency, and the number of the frame (starting from zero and adding one every 100 milliseconds), finally stamped with extension "bmp". Having a series of frames containing relevant information can simplify further analysis such as cell counting or the calculation of the average speed of a cell.

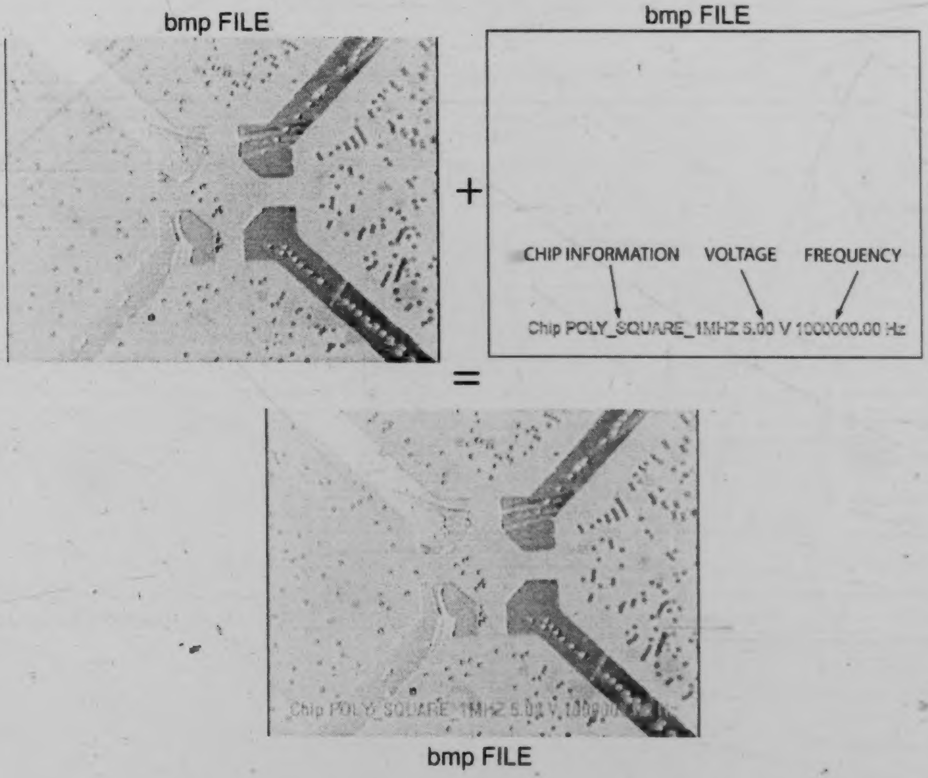

bmp FILE
bmp FILE
+
CHIP INFORMATION
VOLTAGE
FREQUENCY
Chip POLY_SQUARE_1MHZ 5.00 V 1000000.00 Hz
=
Chip POLY_SQUARE_1MHZ 5.00 V 1000000.00 Hz
bmp FILE

Figure 42. Diagram showing the creation of an image with close captioning. A static image containing information about the chip and the voltage and frequency applied to the chip is superimposed upon the video acquired with the video card (connected to a video camera).

Chapter 7

7 FABRICATION OF MICROVALVES AND MICRO-PERISTALTIC PUMPS

7.1 Introduction

Most separation strategies based on dielectrophoresis (as reviewed in the last subsections) included two ports. There are inherent problems to this approach such as clogging the ports due to high cell concentrations, cells possibly remaining at the channels after the separation has taken place leading to false results, and other problems explained in chapter 1. Furthermore, these separations did not have any valves for fluid control. For example, for some DEP applications it would be desired to collect all the cells at specific chamber and then energize the electrodes, but without any lack of control their realization would be difficult and cumbersome. In the future, more collection outlet ports and parallel or massive integration of DEP separation chambers in a microfluidic chip would be required, so control mechanisms to regulate the flow would have to be implemented.

More complex microfluidic channel networks, as well as microfluidic valves and pumps are needed to facilitate the collection of the cells and avoid the problems aforementioned. As discussed in chapter 1, the best approach to implement microfluidic valves and pumps is multilayer soft lithography. In the next section soft lithography is discussed in the context of the design and fabrication of microfluidic valves and microfluidic peristaltic pumps. The following section describes the design of a pneumatic system used to actuate the micropump and the microvalves.

7.2 Multilayer Soft Lithography

Soft lithography is a set of techniques for fabricating microfluidic devices or microstructures for biological applications. The main advantages are rapid prototyping and replica molding. The technique consists of replication of structures patterned in bas-relief on a substrate, 'master' or 'mold'[19, 43, 47, 48, 116].

The master is created by spin coating photoresist (a photosensitive polymer) in a silicon wafer or glass to a desired thickness (5-100 μm). Next, the master is exposed to UV rays below a high resolution transparency film serving as photomask. Positive reliefs are obtained by developing the photoresist. A liquid soft elastomer (usually PDMS) is then poured over the master and heated. The cured soft elastomer replicates the features of the master. The PDMS replica is released from the mold without any damage to the master or the replica. Multiple copies can be cast.

PDMS is supplied in two components, a base and a curing agent. The base contains polydimethylsiloxane bearing vinyl groups and a platinum catalyst; the curing agent contains silicon hydride groups used to cross-link and form covalent bonds with the vinyl groups[44]. They are typically mixed in a ratio of 10:1 (weight/weight) base:curing agent in soft lithography[47].

Multilayer soft lithography is an extension of this methodology. This technique consists of stacking multiple layers of PDMS (replicas can consist of different designs). Having multiple layers allows the creation of active devices or moving parts such as valves and peristaltic pumps [44].

Stacking and bonding of different layers is possible because different concentration ratios of PDMS are used to create every single PDMS layer. Each layer will then have an excess of one of the components, either silicon hydride groups or vinyl groups, as a consequence. Layers coming into contact form a hermetic seal and reactive molecules migrate at the interface formed between two layers [44]. Supplying thermal energy will irreversibly seal the layers together.

7.2.1 Multilayer Soft Lithography Valves and Pumps

Microfluidic pumps [44] and mixers [117] can be formed out of fluidic switches or valves; thus a valve can truly serve as the fundamental component for any microfluidic system just as a transistor is for any integrated circuit [59]. Massive integration of micropumps and microvalves seemed impossible using any hard-material, but multilayer soft lithography has made it possible to design chips with as many as 3574 microvalves and to address 1000 reaction chambers[63] .

Microfluidic valves are created using two PDMS layer (with different concentrations) leaving a thin membrane in between (typically 30 μm). The control layer contains pressure-actuation channels to actuate the microvalves. The network of fluidic channels is located in the flow layer. Each replica containing channels (50μm-300μm wide and 10-50 μm high) are placed one on top of the other forming a cross section, giving birth to a valve.

7.2.2 Push-up and Push-down Valve Configuration

Valves are actuated pneumatically; the principle per se is simple, just imagine stepping into a hose to cut off the flowing of water. Thus, pressure is applied to the control layer in order to deflect the membrane (PDMS) downward or upward (depending on the order of layers), and with sufficient pressure it will close the flow channel. Flow layers situated between the control layer and a glass substrate are called push-down valves (Figure 43.a) because the convex membrane is deflected downwards to close off the channel. Push-up valves deflect the thin and uniform membrane upwards to close off the flow channel; the flow layer is situated on top of the control layer (Figure 43.b). Higher actuation pressures are required on the push-down configuration. Pressures required to drive the micro valves in the push-up valve configuration are one order of magnitude smaller[46]. This difference in pressures arises as a consequence of the differences in the membrane shape, one being convex and the other being uniform and flat. However, both configurations have been proved to work well.

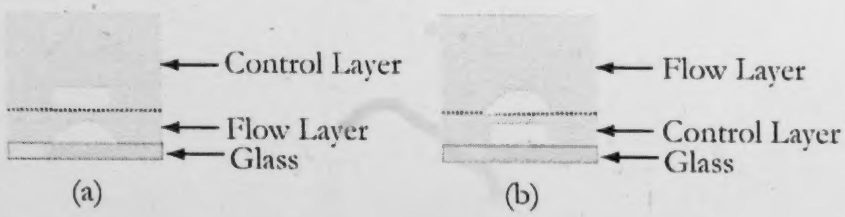


Figure 43. (a) shows a schematic representation of a push-down valve. The flow layer is located underneath the control layer. (b) shows a schematic representation of a push-up valve. The flow layer is located atop the control layer.

7.3 Fabrication of Microvalves and Peristaltic Micropumps

We decided to design and fabricate a simple microfluidic chip consisting of a microfluidic channel, two push-down microvalves and a peristaltic micropump to explore the feasibility of a future integration with DEP-based microfluidic chips. The peristaltic micropump is made by placing three valves in series. The microfluidic channel would be located in the flow layer whereas the other components are located in the control layer. The design is shown in Figure 44; the width of all the channels is 200 microns and the distance separating the valves in the peristaltic pump is 200 microns. Chip designs were drawn in L-Edit ver 8.0 and converted to PS format using LinkCAD Converter ver 5.4 (LinkCAD, CA, USA). The design was sent to Screaming Colour (Edmonton, AB, CA) and printed on plastic transparencies with a high resolution printer (3657 dpi, Fuji Luxel V-9600 CTP, Fuji Photo Film Co., Japan).

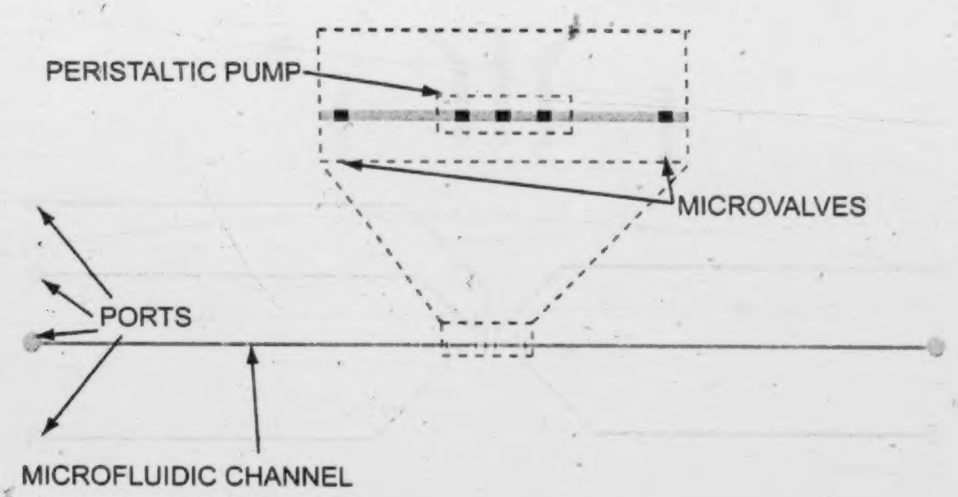


Figure 44. Microfluidic chip design consisting of a microfluidic channel, a peristaltic micropump and two microvalves (not to scale).

In the next sections, we describe the steps to fabricate the microfluidic consisting of two layers of channels with features of 18 microns. The fabrication of two master molds, one for the flow layer and another one for the control layer, is described first. Then, soft lithography replica is described; briefly PDMS is poured over these master molds and partially cured. Next, access ports are punched in the control layer. The control layer is peeled off from the mold and aligned on top of the flow layer. Notice that the flow layer has not been released from the mold. Next, aligned layers are cured completely. Inlet and outlet ports in the flow layer are punched; ports from the flow layer and control layer do not overlap. Finally, the chip is detached from the mold and bonded to a microscope slide or a glass substrate. All fabrication steps have to be performed in sequence and immediately as timing is crucial.

Bigger features can be obtained, as explained in Chapter 4, by double coating a wafer with positive photoresist (SJR5740). Additional steps on the master mold fabrication process would then be needed, such as a second coat, an extra storing day, and probably different times for the photoresist reflow. Another alternative is to use AZ50XT positive photoresist (Clariant Corporation, NJ, USA) which permits 50 micron features in a single coat. But reflow time, exposure and developing time, spin speed and spread speed for the photoresist and the PDMS flow layer would have to be adjusted.

7.3.1 Flow and Control Layer Master Mold Fabrication

This section shows the process flow needed to fabricate microfluidic channels with heights of 20 microns. The fabrication procedure is based on [44, 46]. The features are produced by single-coating a silicon wafer using positive photoresist. Both layers, control and fluidic, are created following the same fabrication procedure. However, the contour of the flow channels must be rounded and an extra step is added so as to reflow the photoresist. This rounded feature will allow a complete deflection of the membrane when the control channel is pressurized, thus allowing the complete sealing of the flow channel.

Four inch silicon wafers were cleaned in a Piranha solution (H_2SO_4:H_2O_2 in a ratio 3:1) for 20 minutes. Wafers are then rinsed with deionized water and put into a Spin Rinse Dryer (SRD, ST-2600D, STI Semitool, MT, USA) for approximately 20 minutes. Next, wafers are dehydrated for 10 minutes at 120°C in an ultraclean remote control radiant heat oven (Labline Imperial III, Labline Instruments Inc, IL, USA) oven. Wafers were left to cool down and then treated with HMDS in a YES HMDS OVEN (Yield Engineering Systems, San Jose, CA, USA) to enhance photoresist adhesion. The HMDS oven was heated up to 150°C and pulled to vacuum until the pressure in the chamber read 1 Torr. The substrates were then exposed for 10 seconds to HMDS vapor. The oven was returned to atmospheric pressure and the substrates were extracted.

Wafers were then placed on a spinner (#5110-CD, Solitec Spinner, CA, USA) configured with the following parameters: spread speed 200 rpm for 20 seconds (cycle 1), spin speed 1000 rpm for 30 seconds (cycle 2). Five milliliters of SRJ5740 ultra-thick photoresist (Shipley Microelectronics, MA, USA) were manually poured on a substrate while it was spinning (cycle 1). The wafers were left atop the chuck for five minutes. Next, a soft-bake step in a preheated oven to 115°C for 10 minutes was performed. The wafers were stored overnight in a box containing a 10 ml beaker of water. The photoresist coating thickness produced was about 18 microns.

A square 5"x5" glass substrate was cleaned in a hot Piranha solution for one hour. Plastic transparencies were then taped to the glass substrates with the printed pattern making contact with the glass substrate. The glass is placed on the mask holder of a mask aligner system (AB-M Inc., Silicon Valley, CA, USA) and brought into contact with the wafers.

Wafers were then UV exposed (λ=365 nm, 18.3 mW/cm2) for 30 seconds. Following this step, Developer 354 (Shipley Microelectronics, MA, USA) was poured in a container and the wafer immersed in it; the container was manually agitated for about 30 minutes, end point determined by visual inspection. Wafers are then rinsed with water. Wafers containing the flow layer were hard-baked in a convection oven at 175°C for 15 minutes to round the flow channels as shown in Figure 45. Photoresist relief thicknesses were characterized with an Alphastep 200 Contact Profilometer (KLA-Tencor, San Jose, CA, USA).

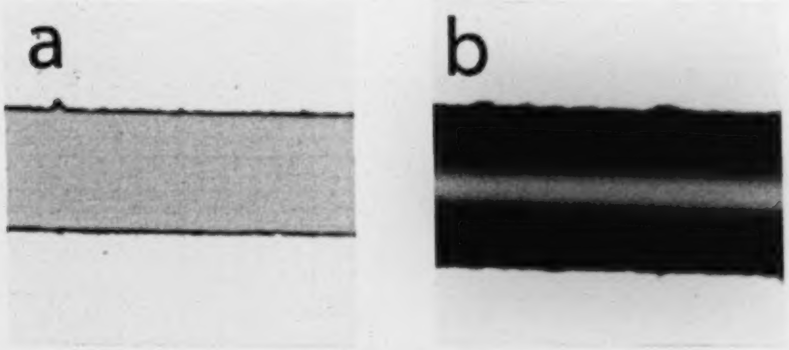


Figure 45. (a) shows a picture of the patterned photoresist on the wafer before soft-bake, features are flat. The picture to the right, (b), shows the same patterned photoresist after it underwent a hard-bake for 15 minutes at 175°C, and shows that the features were rounded.

7.3.2 PDMS Casting

After developing, both wafers were silanized to prevent the adhesion of PDMS to the wafers. Wafers were placed in a desiccator under vacuum with a vial containing a drop of trichloro(1,1,,2,2-perfluoocytl)silane (Sigma-Aldrich, MO, USA) for one hour at room temperature.

Five grams of liquid polymer PDMS (Sylgard 184, Dow Corning) were prepared in a ratio of 20:1 base/curing agent in a 50 g plastic cup and manually stirred with an aluminum bar for 5 minutes. Another fifty grams of PDMS in a ratio of 4:1 were prepared in the same way. Both solutions were degassed for 30 minutes at a room temperature in a PDMS dedicated-oven (Model 1415M, Sheldon Manufacturing Inc, OR, USA) pulled to vacuum (20 in Hg). The same oven was then turned on and set to 80°C.

The wafer containing the flow layer pattern was put in the Solitec spinner. Five grams of PDMS (ratio 20:1) were poured on the spinner while it was spinning at 300 rpm. The spread cycle lasted 25 seconds, and it was followed by a spin cycle at 2200 rpm for 60 seconds. The thickness of the PDMS layer produced was about 28 microns as measured with an Alphastep 200 Contact Profilometer. The wafer was then placed in an aluminum holder.

The other 50 g of PDMS (ratio 4:1) were poured on another metallic holder containing an O-Ring to prevent the PDMS from seeping under the wafer. Both holders were placed in a PDMS dedicated-oven for 40 minutes at 80°C. The metallic holders were extracted from the oven and allowed to cool down for five minutes at room temperature on a metallic table.

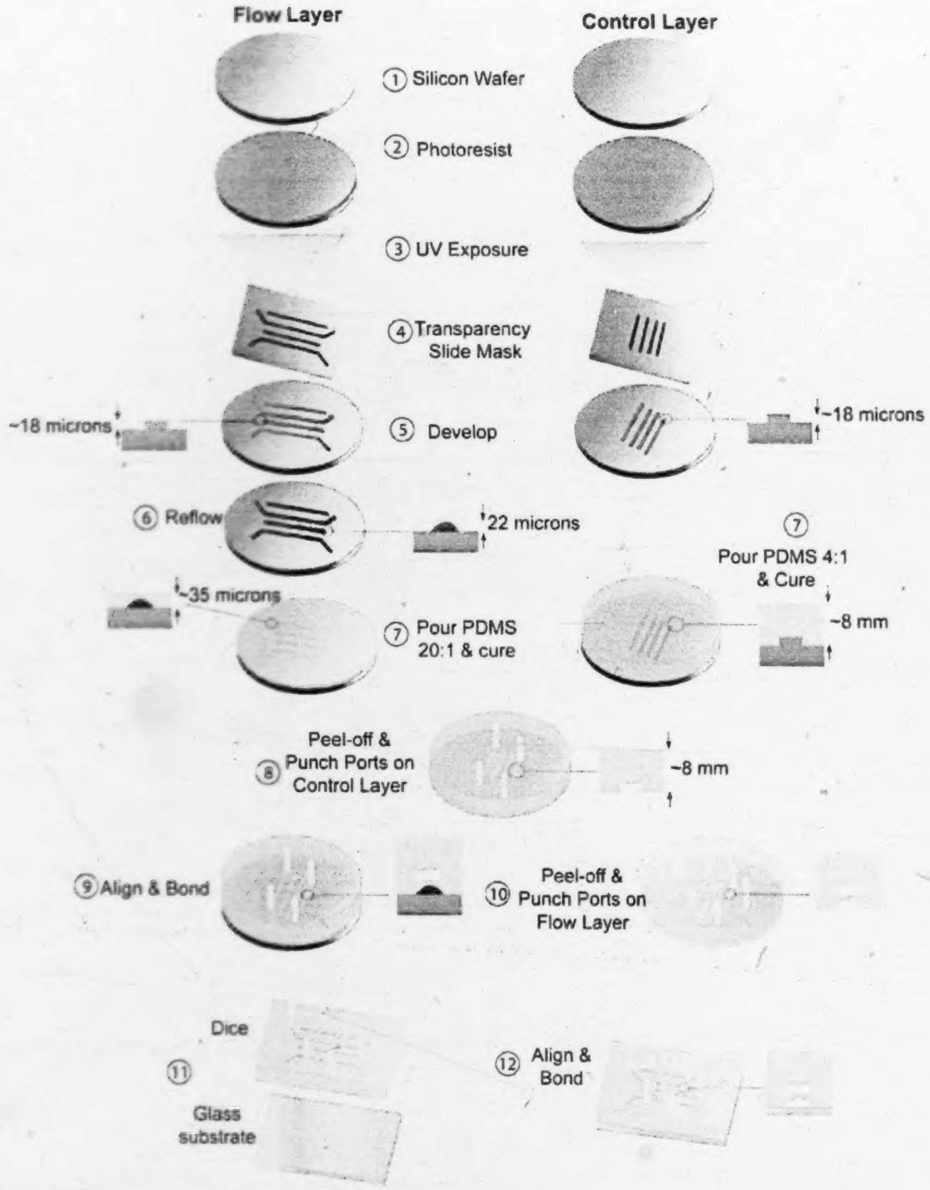

Flow Layer
Control Layer
① Silicon Wafer
② Photoresist
③ UV Exposure
④ Transparency Slide Mask
⑤ Develop
~18 microns
~18 microns
⑥ Reflow
22 microns
⑦ Pour PDMS 4:1 & Cure
~35 microns
⑦ Pour PDMS 20:1 & cure
~8 mm
⑧ Peel-off & Punch Ports on Control Layer
~8 mm
⑨ Align & Bond
⑩ Peel-off & Punch Ports on Flow Layer
Dice
⑪
Glass substrate
⑫ Align & Bond

Figure 46. Schematic representation of the microfluidic chip fabricated by multilayer soft lithography.

A precleaned (as purchased) microscope slide 25x75x1 mm (12-550A, Fisher Scientific, USA) was cleaned in a hot Piranha solution for half an hour. A 20 gauge sterile needle (20 G 1, Becton Dickinson & Co, NJ, USA) was used as a punch. The tip (bevel) of the needle was ground flat prior to use.

The PDMS control layer replica was peeled off from the mold (Figure 46.a), placed in a Petri dish with the features facing upward, and cut out to form a rectangle with dimensions of 25 x 55 mm. Ports were punched with the 20 gauge needle punch. The PDMS replica and the PDMS-coated wafer containing the flow layer were sprayed with ethanol and dried with a N_2 gun. The PDMS replica was then manually aligned on top of the PDMS-coated wafer (Figure 46.b and c).

Next, the wafer containing both replicas was placed again in the metallic holder. Further curing for one and a half hour in the PDMS-dedicated oven at 80°C sealed both layers irreversibly. The metallic holder was extracted from the oven and allowed to cool down for five minutes at room temperature on a metallic table. The PDMS replica was peeled off from the wafer, placed in a Petri dish with the features facing upward, and cut out to form a rectangle with the same dimensions of the PDMS control layer replica. Access ports were punched with the 20 gauge needle punch (Figure 46.d). The microscope slide and the PDMS replica (now being only one) were then sprayed with ethanol and dried with a N_2 gun. The PDMS replica was placed on top of the microscope slide and taken to the PDMS-dedicated oven for further curing at 80°C for three and a half hours (Figure 46.e). The metallic holder was used again to support the microfluidic chip.

Finally, the metallic holder was extracted from the oven and allowed to cool down at room temperature. Pictures of the different components (Figure 46) were taken using a digital camera (E995 COOLPIX, Nikon, Japan) attached to a microscope Leitz (Ergolux, Germany).

Figure 47. Series of pictures showing the components of the microfluidic chip fabricated by multilayer soft lithography: a) micropump, b) microvalve, and c) a punched port.

One foot of Stainless Steel Tubing (0.025 OD x 0.017 ID, Type 304, WD, Full Hard, New England Small Tube Corp, NH, USA) was chopped up in 12 pieces with a length of one inch each of them. Cut out pieces were attached to the punched holes in the microfluidic chip to serve as ports as shown in Figure 48.

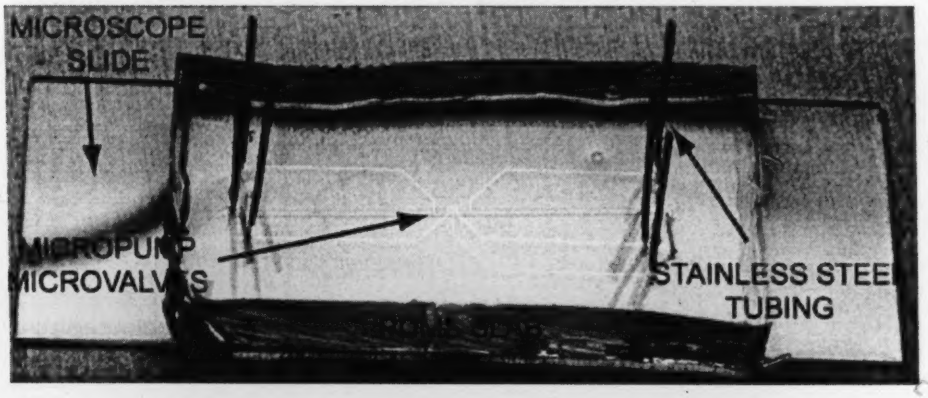

MICROSCOPE
SLIDE
MICROPUMP
MICROVALVES
STAINLESS STEEL
TUBING

Figure 48. Microfluidic chip fabricated by multilayer soft lithography showing the ports (stainless steel tubing), the location of the micropump, the microvalves and the microscope slide.

7.4 Design of a Setup to Control Microvalves and Micropumps

In order to test this microfluidic device, a setup is needed to actuate the microvalves and the micropump. The setup consists of a pneumatic system to supply pressure and control the valves and the pump, and a custom PCB that closes and opens solenoid valves connected to the pneumatic system. The custom PCB is controlled from the computer via the parallel port and consists basically of relays as shown below (Figure 49).

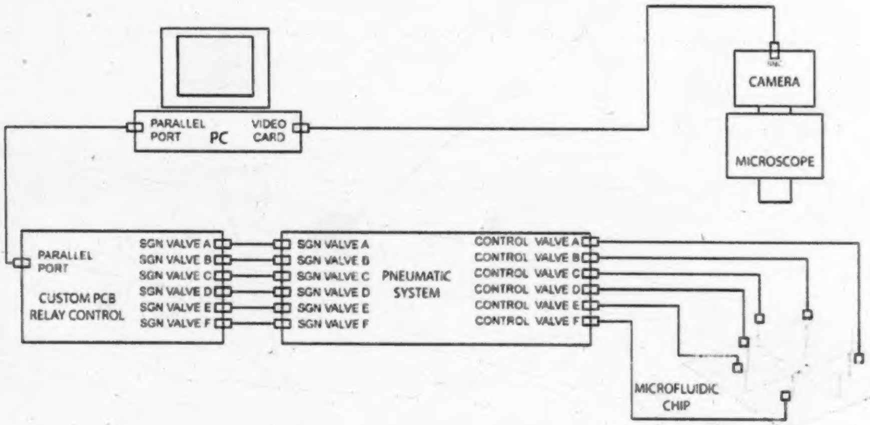
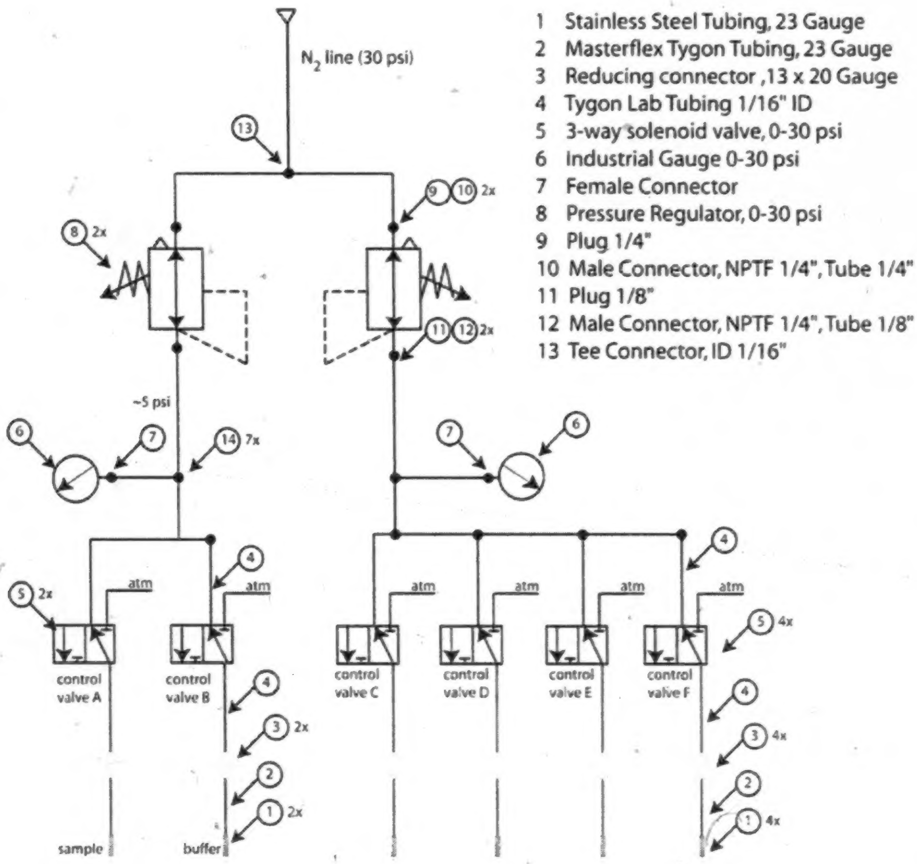

CAMERA
PARALLEL PORT
PC
VIDEO CARD
MICROSCOPE
PARALLEL PORT
CUSTOM PCB RELAY CONTROL
SGN VALVE A
SGN VALVE B
SGN VALVE C
SGN VALVE D
SGN VALVE E
SGN VALVE F
SGN VALVE A
SGN VALVE B
SGN VALVE C
SGN VALVE D
SGN VALVE E
SGN VALVE F
PNEUMATIC SYSTEM
CONTROL VALVE A
CONTROL VALVE B
CONTROL VALVE C
CONTROL VALVE D
CONTROL VALVE E
CONTROL VALVE F
MICROFLUIDIC CHIP

Figure 49. Experimental setup. A custom PCB (controlled through the parallel port of a PC), containing relays and solenoid valves, is connected to a pneumatic system (described below). The microfluidic chip is actuated with the pneumatic system. A camera and a microscope would be needed to observe the actuation of the microvalves.

7.4.1 Pneumatic System

As described before, microvalves and micropumps are actuated with air pressure. Samples would have to be introduced in the microfluidic channels by applying also pressure to the load wells. Pressures needed to inject the solutions into the microfluidic channels are smaller than pressures needed to actuate the microvalves and the micropump. Thus, an external pneumatic system is needed. Furthermore, pressures need to be controlled and read. To this end, we designed a system that consists of two pressure regulators (0 to 30 psi), two pressure gauges, two 2-way solenoid valves, and four 3-way solenoid valves, tees, brass tubes, and hoses in general. A detailed parts list including part numbers and manufactures can be found in Appendix B. Encircled numbers in Figure 50 make reference to a component within the parts list.


1 Stainless Steel Tubing, 23 Gauge
2 Masterflex Tygon Tubing, 23 Gauge
3 Reducing connector ,13 x 20 Gauge
4 Tygon Lab Tubing 1/16" ID
5 3-way solenoid valve, 0-30 psi
6 Industrial Gauge 0-30 psi
7 Female Connector
8 Pressure Regulator, 0-30 psi
9 Plug 1/4"
10 Male Connector, NPTF 1/4", Tube 1/4"
11 Plug 1/8"
12 Male Connector, NPTF 1/4", Tube 1/8"
13 Tee Connector, ID 1/16"

N_2 line (30 psi)
⑬
⑧ 2x
⑨⑩ 2x
⑪⑫ 2x
~5 psi
⑥
⑦
⑭ 7x
⑦
⑥
④
⑤ 2x
atm
atm
atm
atm
atm
atm
⑤ 4x
control valve A
control valve B
control valve C
control valve D
control valve E
control valve F
④
③ 2x
②
① 2x
④
③ 4x
②
① 4x
sample
buffer

Figure 50. Schematic showing the design of a pneumatic system used to supply regulated pressures to the microfluidic chip.

The general purpose pressure line (pressure 40 psi) is split up through a Union Tee (No. 13) into two branches to supply different pressures to the fluidic channels and control channels

As shown in the schematic, the branch to the left connects pressure lines to the microfluidic inlets. It consists of a 30 psi pressure regulator (No. 8, ¼" NPT Port) monitored by a pressure gauge (No. 6, 0 to 30 psi, ¼" NPT port). The pressure line is divided in two lines (No. 13, Tee Connector, 1/16" ID) and connected to two 30 psi 3-way solenoid valves (No. 5, 12 V, ports OD 0.080"). The solenoid valves control the flow through the inlet of the microfluidic channels.

The branch to the right contains also a 30 psi pressure regulator (No. 8, ¼" NPT Port) but four 30 psi 3-way solenoid valves (No.5, 12 V, ports OD 0.080") which are used to open or close the microvalves and the micropump. Four Tee Connectors (No. 13, 1/16" ID) are used to split the pressure line and connect it to the solenoid valves. The normally open port (NO) is connected to atmospheric pressure; whereas, the normally closed port (NC) is connected to the pressurized line. Pressures can be regulated up to 30 psi in both branches.

No. 4 represents Tygon Lab tubing (1/16"ID, 1/18" OD) used to interconnect all the components in the system. No. 9-12 indicate plugs and connectors. No.1 represents a 23-Gauge Stainless steel Hypodermic Tubing which would be plugged into the fluidic and control ports of the chip. Masterflex Tygon tubing (No.2, 0.64mm ID) is slipped on to one end of the needle. Tubing coming from the solenoid valves is connected to the Masterflex tubing through a Reducing Connector (No. 3, 13 x 20 gauge).

7.4.2 Electronic Valve Control

Pneumatic valves are controlled with 6 30-psi High Density Interface (HDI) 3-way solenoid valves (LHDA1233115H, The Lee Company, CT, USA). The HDI 3-way solenoid valves were placed in the custom PCB to facilitate its control. Six 5V Reed relays (8L01-05-011, Coto Technology, RI, USA) are used to operate the valves. Relays are operated through the parallel port of a computer; diodes and transistors in the circuit provide extra protection to the parallel port. Figure 51 shows the electrical circuit used to control the valves. A detailed list of all components can be found in Appendix B. The Printed Circuit Board Layout is included in Appendix D.

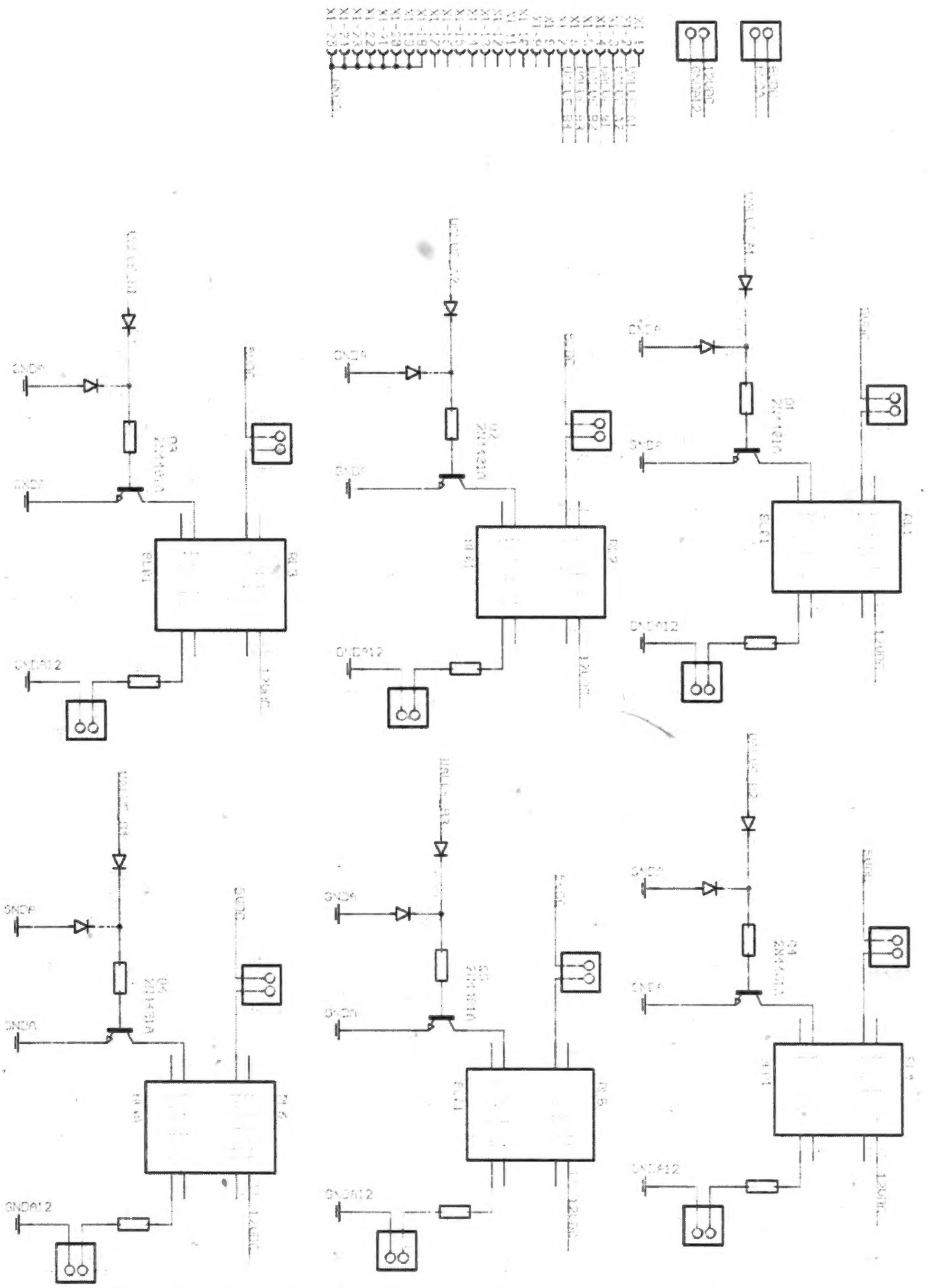


Figure 51. Circuit schematic used to control the three- and two- way valves.
(File Name: relays.sch)

Chapter 8

8 CONCLUSIONS AND FUTURE WORK

8.1 Summary

We have demonstrated the fabrication of DEP devices on PDMS and performed the separation of two different types of latex beads in them. We explained dielectrophoresis and the advantages offered by this technique to the separation and characterization of cells. Although we never worked with cancerous cells, the separation of latex beads exhibiting distinct electrical properties shows that future experiments with human cells is feasible. There is still much debate about the properties of PDMS, although there is an intense research going on in microfluidic components fabricated in PDMS.

We also designed and assembled a custom signal function generator with a cost of US$20 providing almost the same performance as an instrument that costs 10 times more. Software tools to facilitate the analysis of the experiments and to display video on real time with close captioning were also realized.

Microvalves and micropumps fabricated by multilayer soft lithography were also demonstrated. We believe that future microfluidic devices implemented in PDMS would be populated with these valves and pumps. To operate these microfluidic components, a pneumatic system capable of supplying the pressures required to actuate the components was designed. The valves in this pneumatic setup are controlled through a custom PCB containing solenoid valves.

We have thus designed and built microfluidic chips, and an infrastructure for miniaturized DEP that requires only a pressurized air line for operation. This system would be helpful and crucial to the manipulation and separation of cells, the detection of malignant organisms, the characterization of cells, and other applications.

8.2 Future Directions

Microfluidic valves and pumps should be characterized. Future directions would be in the integration of DEP-based microfluidic chips with microfluidic components fabricated by multilayer soft lithography

Complete and accurate characterization of cells can be obtained from electrorotation spectrum graphs or from the DEP velocity spectrum as explained in chapter 2. Our software tools explained in Chapter 6 would diminish considerably the time to obtain these spectra. Furthermore, software tools (for example MATLAB 7, MathWorks, M, USA) nowadays allow for counting population of cells and to track individual cells within a mixture of cells so it will also permit rapid identification of cells based on morphology and size.

It is clear that any DEP separation strategy would have to integrate a mechanism to control or regulate the flow. It is also evident from the material discussed in this thesis that any DEP-cell sorter microdevice has to be built out of the same material. The simplest mechanism to control flow is to integrate multilayer soft lithography micro valves and pumps into the device. These valves could be introduced to solve this issue since multiple electrode configurations powered at different voltage and frequencies could be integrated in one chip and valves could aid in the concentration of cells at defined areas.

A future separation strategy had to include two outlets, one for the collection of the target cells and the other for waste or non desired particles. Switch valves are needed to facilitate the collection of the cells and avoid the problems aforementioned. In the future, a design like this would permit the integration with other types of assays such as PCR.

The DEP-separation chip would consist of three layers. Gold patterned electrodes on glass containing a specific DEP electro configuration would form the first layer. The two other layers are made out of PDMS; one of them contains the flow channels whereas the other houses the control lines. Four microvalves to control the flow through the channels are implemented in the control layer and two channels forming a cross section is embedded in the flow layer. Electrodes are located at this intersection as shown in Figure 52. Two inlets are punched in the flow layer, one to supply the solution and the other to supply the separation buffer; another two holes are perforated to function as outlets, one to collect the waste and the other to collect the particles of interest as shown in Figure 53.

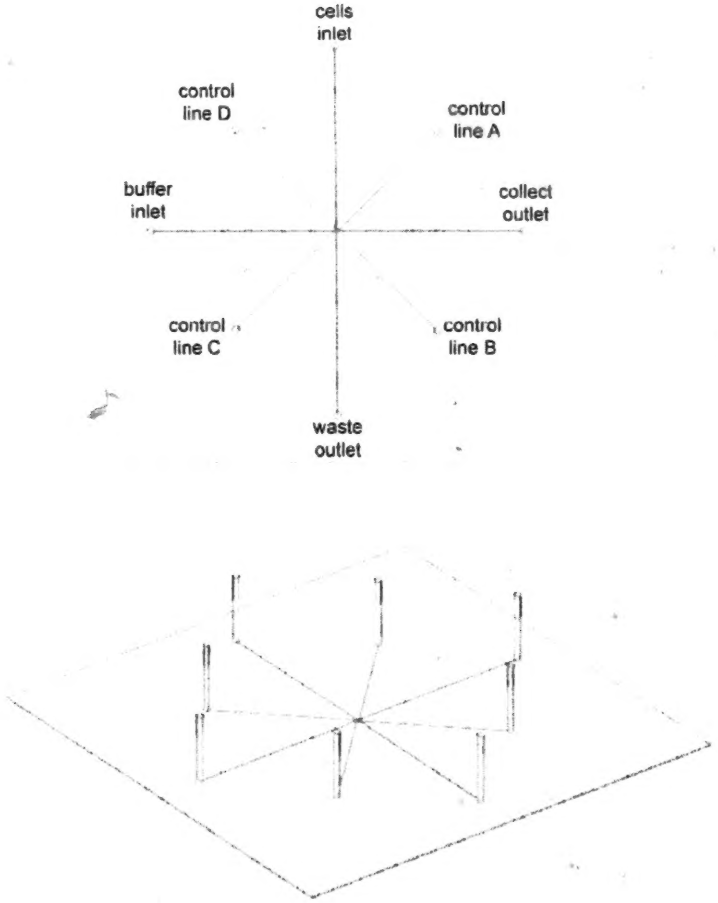

cells
inlet
control
line D
control
line A
buffer
inlet
collect
outlet
control
line C
control
line B
waste
outlet

Figure 52. Top view (left) and cross sectional view of the Cell DEP Capture Device showing microvalves, flow channels and electrodes, with dimensions.

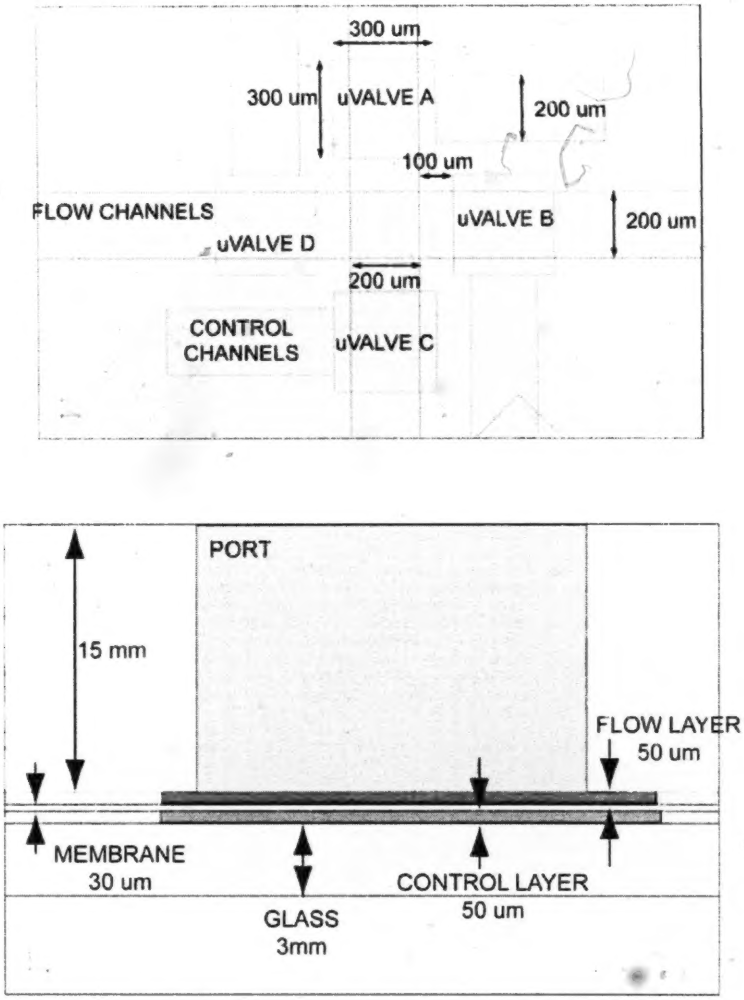

300 um
300 um
uVALVE A
200 um
100 um
FLOW CHANNELS
uVALVE B
200 um
uVALVE D
200 um
CONTROL CHANNELS
uVALVE C
PORT
15 mm
FLOW LAYER 50 um
MEMBRANE 30 um
CONTROL LAYER 50 um
GLASS 3mm

Figure 53. Top view (left) and 3D view showing the control lines implemented in the control layer and the flow channels located in the fluidic layer.

The fabrication protocol was already discussed in the previous chapter. The pneumatic system described previously can also be used to actuate the valves and to inject the samples.

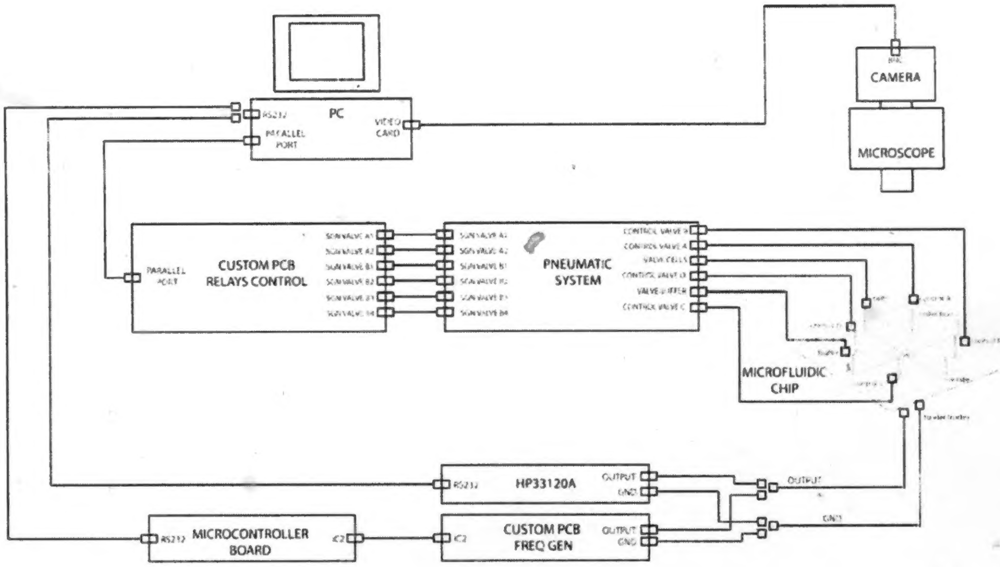

CAMERA
MICROSCOPE
PC
RS232
PARALLEL PORT
VIDEO CARD
PARALLEL PORT
CUSTOM PCB RELAYS CONTROL
SGN VALVE A1
SGN VALVE A2
SGN VALVE B1
SGN VALVE B2
SGN VALVE B3
SGN VALVE B4
PNEUMATIC SYSTEM
CONTROL VALVE B
CONTROL VALVE A
VALVE CELLS
CONTROL VALVE D
VALVE BUFFER
CONTROL VALVE C
MICROFLUIDIC CHIP
RS232
HP33120A
OUTPUT
GND
OUTPUT
GND
RS232
MICROCONTROLLER BOARD
I2C
I2C
CUSTOM PCB FREQ GEN
OUTPUT
GND

Figure 54. A possible setup to drive DEP-based microfluidic chips integrated with microfluidic valves

The protocol of operation would be simple. Initially, all inlets and outlets are exposed to atmospheric pressure and the microvalves are closed. The solution containing the cells and buffer are loaded at the cell and buffer inlets, respectively; pressure lines are then connected to both inlet ports. Microvalves A and C are open, while the rest remain closed; pressure is applied to the cell inlet port by enabling control valve S. At the same time electrodes are energized so the dielectrophoretic separation process starts taking place. Fluid then starts to move along the channel.

Cells of interest must have been trapped at the electrodes or eluted at the outlet port and collected there. Given that positive and negative dielectrophoresis can be used to trap cells, the collection and waste outlet ports are used interchangeably. In this case, we assume that positive dielectrophoresis is the trapping mechanism.

Cells collected at the electrodes can then be eluted by closing microvalves A and C, and opening microvalves B and D and enabling control valve B to push the buffer through the channel.

Bibliography

1. Huang, Y., E.L. Mather, J.L. Bell, and M. Madou, *MEMS-based sample preparation for molecular diagnostics.* Analytical and Bioanalytical Chemistry, 2002. **372**(1): p. 49-65.

2. Gascoyne, P.R.C. and J.V. Vykoukal, *Dielectrophoresis-based sample handling in general-purpose programmable diagnostic instruments,* Proceedings of the Ieee, 2004. **92**(1): p. 22-42.

3. Wang, X.B., J. Yang, Y. Huang, J. Vykoukal, F.F. Becker, and P.R.C. Gascoyne, *Cell separation by dielectrophoretic field-flow-fractionation.* Analytical Chemistry, 2000. **72**(4): p. 832-839.

4. Dolnik, V., S. Liu, and S. Jovanovich, *Capillary electrophoresis on microchip.* Electrophoresis, 2000. **21**(1): p. 41-54.

5. Barman B. N., Ashwood E. R., and Giddings J. C., *Separation and Size Distribution of Red Blood Cells of Diverse Size, Shape, and Origin by Flow/Hyperlayer Field-Flow Fractionation.* Analytical Biochemistry, 1993. **212**(1): p. 35-42.

6. Chianea, T., N.E. Assidjo, and P.J.P. Cardot, *Sedimentation field-flow-fractionation: emergence of a new cell separation methodology.* Talanta, 2000. **51**(5): p. 835-847.

7. Giddings, J.C., *Field-Flow Fractionation: Analysis of Macromolecular, Colloidal, and Particulate Materials.* Science, 1993. **260**(5113): p. 1456-1465.

8. Parsons, R., V. Yue, X. Tong, P. Cardot, A. Bernard, J.P. Andreux, and K. Caldwell, *Comparative study of human red blood cell analysis with three different field-flow fractionation systems.* Journal of Chromatography B: Biomedical Sciences and Applications, 1996. **686**(2): p. 177-187.

9. [Anon], *Flow Field Flow Fractionation, a Versatile Novel Separation.* Naturwissenschaften, 1977. **64**(3): p. 143-143.

10. Colfen, H. and M. Antonietti, *Field-flow fractionation techniques for polymer and colloid analysis.* New Developments in Polymer Analytics I, 2000. **150**: p. 67-187.

11. Gascoyne, P.R.C. and J. Vykoukal, *Particle separation by dielectrophoresis.* Electrophoresis, 2002. **23**(13): p. 1973-1983.

12. Voisard, D., F. Meuwly, P.A. Ruffieux, G. Baer, and A. Kadouri, *Potential of cell retention techniques for large-scale high-density perfusion culture of suspended mammalian cells.* Biotechnol Bioeng, 2003. **82**(7): p. 751-65.

13. Wickramasinghe, S.R., B. Han, J.O. Carlson, and S.M. Powers, *Clearance of minute virus of mice by flocculation and microfiltration.* Biotechnol Bioeng, 2004. **86**(6): p. 612-21.

14. Wilding, P., L.J. Kricka, J. Cheng, G. Hvichia, M.A. Shoffner, and P. Fortina, *Integrated cell isolation and polymerase chain reaction analysis using silicon microfilter chambers.* Anal Biochem, 1998. **257**(2): p. 95-100.

15. Herzenberg, L.A., D. Parks, B. Sahaf, O. Perez, and M. Roederer, *The history and future of the fluorescence activated cell sorter and flow cytometry: a view from Stanford.* Clin Chem, 2002. **48**(10): p. 1819-27.

16. Fu, A.Y., C. Spence, A. Scherer, F.H. Arnold, and S.R. Quake, *A microfabricated fluorescence-activated cell sorter.* Nature Biotechnology, 1999. **17**(11): p. 1109-1111.

17. Fu, A.Y., H.P. Chou, C. Spence, F.H. Arnold, and S.R. Quake, *An integrated microfabricated cell sorter.* Analytical Chemistry, 2002. **74**(11): p. 2451-2457.

18. Deng, T., M. Prentiss, and G.M. Whitesides, *Fabrication of magnetic microfiltration systems using soft lithography.* Applied Physics Letters, 2002. **80**(3): p. 461-463.

19. Sia, S.K. and G.M. Whitesides, *Microfluidic devices fabricated in poly(dimethylsiloxane) for biological studies.* Electrophoresis, 2003. **24**(21): p. 3563-76.

20. Becker, F.F., X.B. Wang, Y. Huang, R. Pethig, J. Vykoukal, and P.R.C. Gascoyne, *The Removal of Human Leukemia-Cells from Blood Using Interdigitated Microelectrodes.* Journal of Physics D-Applied Physics, 1994. **27**(12): p. 2659-2662.

21. Goater, A.D. and R. Pethig, *Electrorotation and dielectrophoresis.* Parasitology, 1998. **117**: p. S177-S189.

22. Heida, T., W.L.C. Rutten, and E. Marani, *Understanding dielectrophoretic trapping of neuronal cells: modelling electric field, electrode-liquid interface and fluid flow.* Journal of Physics D-Applied Physics, 2002. **35**(13): p. 1592-1602.

23. Docoslis, A. and P. Alexandridis, *One-, two-, and three-dimensional organization of colloidal particles using nonuniform alternating current electric fields.* Electrophoresis, 2002. **23**(14): p. 2174-2183.

24. Voldman, J., M.L. Gray, M. Toner, and M.A. Schmidt, *A microfabrication-based dynamic array cytometer.* Analytical Chemistry, 2002. **74**(16): p. 3984-3990.

25. Schnelle, T., T. Muller, G. Gradl, S.G. Shirley, and G. Fuhr, *Dielectrophoretic manipulation of suspended submicron particles.* Electrophoresis, 2000. **21**(1): p. 66-73.

26. Voldman, J., R.A. Braff, M. Toner, M.L. Gray, and M.A. Schmidt, *Holding forces of single-particle dielectrophoretic traps.* Biophysical Journal, 2001. **80**(1): p. 531-541.

27. Chou, C.F., J.O. Tegenfeldt, O. Bakajin, S.S. Chan, E.C. Cox, N. Darnton, T. Duke, and R.H. Austin, *Electrodeless dielectrophoresis of single- and double-stranded DNA.* Biophysical Journal, 2002. **83**(4): p. 2170-2179.

28. Gascoyne, P.R.C., Y. Huang, R. Pethig, J. Vykoukal, and F.F. Becker, *Dielectrophoretic Separation of Mammalian-Cells Studied by Computerized Image-Analysis.* Measurement Science & Technology, 1992. **3**(5): p. 439-445.

29. Hughes, M.P., *Micro- and nano-electrokinetics in medicine.* Engineering in Medicine and Biology Magazine, IEEE, 2003. **22**(6): p. 32-42.

30. Pethig, R., M.S. Talary, and R.S. Lee, *Enhancing traveling-wave dielectrophoresis with signal superposition.* Engineering in Medicine and Biology Magazine, IEEE, 2003. **22**(6): p. 43-50.

31. Wang, X.B., Y. Huang, P.R.C. Gascoyne, and F.F. Becker, *Dielectrophoretic manipulation of particles.* Ieee Transactions on Industry Applications, 1997. **33**(3): p. 660-669.

32. Muller, T., A. Pfennig, P. Klein, G. Gradl, M. Jager, and T. Schnelle, *The potential of dielectrophoresis for single-cell experiments.* Engineering in Medicine and Biology Magazine, IEEE, 2003. **22**(6): p. 51-61.

33. Medoro, G., N. Manaresi, A. Leonardi, L. Altomare, M. Tartagni, and R. Guerrieri, *A lab-on-a-chip for cell detection and manipulation.* Ieee Sensors Journal, 2003. **3**(3): p. 317-325.

34. Huang, Y. and R. Pethig, *Electrode Design for Negative Dielectrophoresis.* Measurement Science & Technology, 1991. **2**(12): p. 1142-1146.

35. Wang, X.B., M.P. Hughes, Y. Huang, F.F. Becker, and P.R.C. Gascoyne, *Nonuniform Spatial Distributions of Both the Magnitude and Phase of Ac Electric-Fields Determine Dielectrophoretic Forces.* Biochimica Et Biophysica Acta-General Subjects, 1995. **1243**(2): p. 185-194.

36. Green, N.G., A. Ramos, A. Gonzalez, H. Morgan, and A. Castellanos, *Fluid flow induced by nonuniform ac electric fields in electrolytes on microelectrodes. III. Observation of streamlines and numerical simulation.* Physical Review E, 2002. **66**(2)

37. Hoettges, K.F., M.B. McDonnell, and M.P. Hughes, *Use of combined dielectrophoretic/electrohydrodynamic forces for biosensor enhancement.* Journal of Physics D-Applied Physics, 2003. **36**(20): p. L101-L104.

38. Kopp, M.U., H.J. Crabtree, and A. Manz, *Developments in technology and applications of microsystems.* Curr Opin Chem Biol, 1997. **1**(3): p. 410-9.

39. Bousse, L., C. Cohen, T. Nikiforov, A. Chow, A.R. Kopf-Sill, R. Dubrow, and J.W. Parce, *Electrokinetically controlled microfluidic analysis systems.* Annu Rev Biophys Biomol Struct, 2000. **29**: p. 155-81.

40. Hughes, M.P., *Strategies for dielectrophoretic separation in laboratory-on-a-chip systems.* Electrophoresis, 2002. **23**(16): p. 2569-82.

41. Voldman, J., M.L. Gray, and M.A. Schmidt, *Microfabrication in biology and medicine.* Annu Rev Biomed Eng, 1999. **1**: p. 401-25.

42. Quake, S.R. and A. Scherer, *From micro- to nanofabrication with soft materials.* Science, 2000. **290**(5496): p. 1536-1540.

43. McDonald, J.C. and G.M. Whitesides, *Poly(dimethylsiloxane) as a material for fabricating microfluidic devices.* Acc Chem Res, 2002. **35**(7): p. 491-9.

44. Unger, M.A., H.P. Chou, T. Thorsen, A. Scherer, and S.R. Quake, *Monolithic microfabricated valves and pumps by multilayer soft lithography.* Science, 2000. **288**(5463): p. 113-116.

45. Watson, J.M. and M.G. Baron, *The behaviour of water in poly(dimethylsiloxane).* Journal of Membrane Science, 1996. **110**(1): p. 47-57.

46. Studer, V., G. Hang, A. Pandolfi, M. Ortiz, W.F. Anderson, and S.R. Quake, *Scaling properties of a low-actuation pressure microfluidic valve.* Journal of Applied Physics, 2004. **95**(1): p. 393-398.

47. McDonald, J.C., D.C. Duffy, J.R. Anderson, D.T. Chiu, H. Wu, O.J. Schueller, and G.M. Whitesides, *Fabrication of microfluidic systems in poly(dimethylsiloxane).* Electrophoresis, 2000. **21**(1): p. 27-40.

48. Ng, J.M., I. Gitlin, A.D. Stroock, and G.M. Whitesides, *Components for integrated poly(dimethylsiloxane) microfluidic systems.* Electrophoresis, 2002. **23**(20): p. 3461-73.

49. Lotters, J.C., W. Olthuis, P.H. Veltink, and P. Bergveld, *The mechanical properties of the rubber elastic polymer polydimethylsiloxane for sensor applications.* Journal of Micromechanics and Microengineering, 1997. **7**(3): p. 145-147.

50. Kim, J., M.K. Chaudhury, and M.J. Owen, *Hydrophobicity loss and recovery of silicone HV insulation.* Ieee Transactions on Dielectrics and Electrical Insulation, 1999. **6**(5): p. 695-702.

51. Blume, I., P.J.F. Schwering, M.H.V. Mulder, and C.A. Smolders, *Vapor Sorption and Permeation Properties of Poly(Dimethylsiloxane) Films.* Journal of Membrane Science, 1991. **61**: p. 85-97.

52. Reyes, D.R., D. Iossifidis, P.A. Auroux, and A. Manz, *Micro total analysis systems. 1. Introduction, theory, and technology.* Anal Chem, 2002. **74**(12): p. 2623-36.

53. Beebe, D.J., G.A. Mensing, and G.M. Walker, *Physics and applications of microfluidics in biology.* Annual Review of Biomedical Engineering, 2002. **4**: p. 261-286.

54. Nguyen, N.T., X.Y. Huang, and T.K. Chuan, *MEMS-micropumps: A review.* Journal of Fluids Engineering-Transactions of the Asme, 2002. **124**(2): p. 384-392.

55. Jakeway, S.C., A.J. de Mello, and E.L. Russell, *Miniaturized total analysis systems for biological analysis.* Fresenius J Anal Chem, 2000. **366**(6-7): p. 525-39.

56. Verpoorte, E. and N.F. De Rooij, *Microfluidics meets MEMS.* Proceedings of the Ieee, 2003. **91**(6): p. 930-953.

57. Bart, S.F., L.S. Tavrow, M. Mehregany, and J.H. Lang, *Microfabricated Electrohydrodynamic Pumps.* Sensors and Actuators a-Physical, 1990. **21**(1-3): p. 193-197.

58. Felton, M.J., *The new generation of microvalves.* Analytical Chemistry, 2003. **75**(19): p. 429a-432a.

59. Hansen, C. and S.R. Quake, *Microfluidics in structural biology: smaller, faster... better.* Current Opinion in Structural Biology, 2003. **13**(5): p. 538-544.

60. Liu, R.H., Q. Yu, and D.J. Beebe, *Fabrication and characterization of hydrogel-based microvalves.* Microelectromechanical Systems, Journal of, 2002. **11**(1): p. 45-53.

61. Baldi, A., Y. Gu, P.E. Loftness, R.A. Siegel, and B. Ziaie, *A hydrogel-actuated environmentally sensitive microvalve for active flow control.* Microelectromechanical Systems, Journal of, 2003. **12**(5): p. 613-621.

62. Eddington, D.T. and D.J. Beebe, *A valved responsive hydrogel microdispensing device with integrated pressure source.* Microelectromechanical Systems, Journal of, 2004. **13**(4): p. 586-593.

63. Thorsen, T., S.J. Maerkl, and S.R. Quake, *Microfluidic large-scale integration.* Science, 2002. **298**(5593): p. 580-584.

64. Durr, M., J. Kentsch, T. Muller, T. Schnelle, and M. Stelzle, *Microdevices for manipulation and accumulation of micro- and nanoparticles by dielectrophoresis.* Electrophoresis, 2003. **24**(4): p. 722-31.

65. Falokun, C.D., F. Mavituna, and G.H. Markx, *AC electrokinetic characterisation and separation of cells with high and low embryogenic potential in suspension cultures of carrot (Daucus carota).* Plant Cell Tissue and Organ Culture, 2003. **75**(3): p. 261-272.

66. Minerick, A.R., R.H. Zhou, P. Takhistov, and H.C. Chang, *Manipulation and characterization of red blood cells with alternating current fields in microdevices.* Electrophoresis, 2003. **24**(21): p. 3703-3717.

67. Auerswald, J. and H.F. Knapp, *Quantitative assessment of dielectrophoresis as a micro fluidic retention and separation technique for beads and human blood erythrocytes.* Microelectronic Engineering, 2003. **67-8**: p. 879-886.

68. Suehiro, J., D. Noutomi, M. Shutou, and M. Hara, *Selective detection of specific bacteria using dielectrophoretic impedance measurement method combined with an antigen-antibody reaction.* Journal of Electrostatics, 2003. **58**(3-4): p. 229-246.

69. Kumar, A., Z. Qiu, A. Acrivos, B. Khusid, and D. Jacqmin, *Combined negative dielectrophoresis and phase separation in nondilute suspensions subject to a high-gradient ac electric field.* Phys Rev E Stat Nonlin Soft Matter Phys, 2004. **69**(2): p. 021402.

70. Holmes, D., N.G. Green, and H. Morgan, *Microdevices for dielectrophoretic flow-through cell separation.* Engineering in Medicine and Biology Magazine, IEEE, 2003. **22**(6): p. 85-90.

71. Hoettges, K.F., M.P. Hughes, A. Cotton, N.A.E. Hopkins, and M.B. McDonnell, *Optimizing particle collection for enhanced surface-based biosensors.* Engineering in Medicine and Biology Magazine, IEEE, 2003. **22**(6): p. 68-74.

72. Griffiths, D.J., *Introduction to electrodynamics.* 2 ed. 1989, New Jersey, USA: Prentice-Hall. p. 532.

73. Von Hippel, A.R., *Dielectric and waves.* 1 ed. 1954, New York, USA: John Wiley & Sons. p. 284.

74. Pohl, H.A., *Dielectrophoresis, the behaviour of neutral matter in nonuniform electric fields.* 1978, Cambridge, USA: Cambridge University Press.

75. Wang, X.-B., Y. Huang, R. Holzel, J.P.H. Burt, and R. Pethig, *Theoretical and experimental investigations of the interdependence of the dielectric, dielectrophoretic and electrorotational behaviour of colloidal particles.* Journal of Physics D: Applied Physics, 1993. **26**(2): p. 312.

76. Huang, Y., R. Holzel, R. Pethig, and X.B. Wang, *Differences in the AC electrodynamics of viable and non-viable yeast cells determined through combined dielectrophoresis and electrorotation studies.* Phys Med Biol, 1992. **37**(7): p. 1499-517.

77. Huang, Y., X.B. Wang, R. Holzel, F.F. Becker, and P.R.C. Gascoyne, *Electrorotational Studies of the Cytoplasmic Dielectric-Properties of Friend Murine Erythroleukemia-Cells.* Physics in Medicine and Biology, 1995. **40**(11): p. 1789-1806.

78. Price, J.A.R., J.P.H. Burt, and R. Pethig, *Applications of a New Optical Technique for Measuring the Dielectrophoretic Behavior of Microorganisms.* Biochimica Et Biophysica Acta, 1988. **964**(2): p. 221-230.

79. Burt, J.P.H., T.A.K. Alameen, and R. Pethig, *An Optical Dielectrophoresis Spectrometer for Low-Frequency Measurements on Colloidal Suspensions.* Journal of Physics E-Scientific Instruments, 1989. **22**(11): p. 952-957.

80. Albert, B., D. Bray, J. Lewis, M. Raff, K. Roberts, and J. Watson, *Molecular Biology of the Cell.* 1994, NY, USA: Garland Publishing.

81. Irimajiri, A., T. Hanai, and A. Inouye, *A dielectric theory of "multi-stratified shell" model with its application to a lymphoma cell.* J Theor Biol, 1979. **78**(2): p. 251-69.

82. Fishman, H., *Cell Membrane Structure and Function*, in *Interactions between electromagnetic fields and cells*, A. Chiabrera, C. Nicolini, and H.P. Schwan, Editors. 1985, Plenum Press: NY, USA.

83. Chan, K.L., P.R.C. Gascoyne, F.F. Becker, and R. Pethig, *Electrorotation of liposomes: verification of dielectric multi-shell model for cells.* Biochimica Et Biophysica Acta-Lipids and Lipid Metabolism, 1997. **1349**(2): p. 182-196.

84. Wang, X.B., Y. Huang, J.P.H. Burt, G.H. Markx, and R. Pethig, *Selective Dielectrophoretic Confinement of Bioparticles in Potential-Energy Wells.* Journal of Physics D-Applied Physics, 1993. **26**(8): p. 1278-1285.

85. Ying, H., R. Holzel, R. Pethig, and X.B. Wang, *Differences in the Ac Electrodynamics of Viable and Nonviable Yeast-Cells Determined through Combined Dielectrophoresis and Electrorotation Studies.* Physics in Medicine and Biology, 1992. **37**(7): p. 1499-1517.

86. Huang, Y., X.B. Wang, J.A. Tame, and R. Pethig, *Electrokinetic Behavior of Colloidal Particles in Traveling Electric-Fields - Studies Using Yeast-Cells.* Journal of Physics D-Applied Physics, 1993. **26**(9): p. 1528-1535.

87. Gascoyne, P., C. Mahidol, M. Ruchirawat, J. Satayavivad, P. Watcharasit, and F.F. Becker, *Microsample preparation by dielectrophoresis: isolation of malaria.* Lab on a Chip, 2002. **2**(2): p. 70-75.

88. Li, H.B. and R. Bashir, *Dielectrophoretic separation and manipulation of live and heat-treated cells of Listeria on microfabricated devices with interdigitated electrodes.* Sensors and Actuators B-Chemical, 2002. **86**(2-3): p. 215-221.

89. Gascoyne, P.R.C., J. Noshari, F.F. Becker, and R. Pethig, *Use of dielectrophoretic collection spectra for characterizing differences between normal and cancerous cells.* Industry Applications, IEEE Transactions on, 1994. **30**(4): p. 829-834.

90. Huang, Y., J.M. Yang, P.J. Hopkins, S. Kassegne, M. Tirado, A.H. Forster, and H. Reese, *Separation of simulants of biological warfare agents from blood by a miniaturized dielectrophoresis device.* Biomedical Microdevices, 2003. **5**(3): p. 217-225.

91. Markarian, N., M. Yeksel, B. Khusid, K. Farmer, and A. Acrivos, *Limitations on the scale of an electrode array for trapping particles in microfluidics by positive dielectrophoresis.* Applied Physics Letters, 2003. **82**(26): p. 4839-4841.

92. Xia, Y.N. and G.M. Whitesides, *Soft lithography.* Angewandte Chemie-International Edition, 1998. **37**(5): p. 551-575.

93. Han, A., M. Graff, O. Wang, and A.B. Frazier. *A multi-layer microfluidic system with integrated electrical and on-column optical detection.* in *Sensors, 2002. Proceedings of IEEE.* 2002.

94. Chen, H., D. Acharya, A. Gajraj, and J.C. Meiners, *Robust interconnects and packaging for microfluidic elastomeric chips.* Analytical Chemistry, 2003. **75**(19): p. 5287-5291.

95. Li, S.F. and S.C. Chen, *Polydimethylsioxane fluidic interconnects for microfluidic systems.* Ieee Transactions on Advanced Packaging, 2003. **26**(3): p. 242-247.

96. Enikov, E.T. and J.G. Boyd, *Electroplated electro-fluidic interconnects for chemical sensors.* Sensors and Actuators a-Physical, 2000. **84**(1-2): p. 161-164.

97. Gray, B.L., D. Jaeggi, N.J. Mourlas, B.P. van Drieenhuizen, K.R. Williams, N.I. Maluf, and G.T.A. Kovacs, *Novel interconnection technologies for integrated microfluidic systems.* Sensors and Actuators a-Physical, 1999. **77**(1): p. 57-65.

98. Gonzalez, C., S.D. Collins, and R.L. Smith, *Fluidic interconnects for modular assembly of chemical microsystems.* Sensors and Actuators B-Chemical, 1998. **49**(1-2): p. 40-45.

99. Systems, D., *SLA 7000 Systems.* 2004.

100. Jo, B.H., L.M. Van Lerberghe, K.M. Motsegood, and D.J. Beebe, *Three-dimensional micro-channel fabrication in polydimethylsiloxane (PDMS) elastomer.* Journal of Microelectromechanical Systems, 2000. **9**(1): p. 76-81.

101. Scientific, U., *Micro Fluidic connections.* 2004.

102. Li, S. and S. Chen, *Polydimethylsioxane fluidic interconnects for microfluidic systems.* Advanced Packaging, IEEE Transactions on [see also Components, Packaging and Manufacturing Technology, Part B: Advanced Packaging, IEEE Transactions on], 2003. **26**(3): p. 242-247.

103. Duffy, D.C., J.C. McDonald, O.J.A. Schueller, and G.M. Whitesides, *Rapid prototyping of microfluidic systems in poly(dimethylsiloxane).* Analytical Chemistry, 1998. **70**(23): p. 4974-4984.

104. Kricka, L.J. and P. Wilding, *Microchip PCR.* Analytical and Bioanalytical Chemistry, 2003. **377**(5): p. 820-825.

105. Erill, I., S. Campoy, N. Erill, J. Barbe, and J. Aguilo, *Biochemical analysis and optimization of inhibition and adsorption phenomena in glass-silicon PCR-chips.* Sensors and Actuators B-Chemical, 2003. **96**(3): p. 685-692.

106. Lee, J.N., C. Park, and G.M. Whitesides, *Solvent compatibility of poly(dimethylsiloxane)-based microfluidic devices.* Anal Chem, 2003. **75**(23): p. 6544-54.

107. Makamba, H., J.H. Kim, K. Lim, N. Park, and J.H. Hahn, *Surface modification of poly(dimethylsiloxane) microchannels.* Electrophoresis, 2003. **24**(21): p. 3607-19.

108. Ren, X., M. Bachman, C. Sims, G.P. Li, and N. Allbritton, *Electroosmotic properties of microfluidic channels composed of poly(dimethylsiloxane).* J Chromatogr B Biomed Sci Appl, 2001. **762**(2): p. 117-25.

109. Spehar, A.M., S. Koster, V. Linder, S. Kulmala, N.F. de Rooij, E. Verpoorte, H. Sigrist, and W. Thormann, *Electrokinetic characterization of poly(dimethylsiloxane) microchannels.* Electrophoresis, 2003. **24**(21): p. 3674-8.

110. Leach, A.M., A.R. Wheeler, and R.N. Zare, *Flow injection analysis in a microfluidic format.* Anal Chem, 2003. **75**(4): p. 967-72.

111. Matsubara, Y., Y. Murakami, M. Kobayashi, Y. Morita, and E. Tamiya, *Application of on-chip cell cultures for the detection of allergic response.* Biosens Bioelectron, 2004. **19**(7): p. 741-7.

112. Hong, J.W., T. Fujii, M. Seki, T. Yamamoto, and I. Endo. *PDMS (polydimethylsiloxane)-glass hybrid microchip for gene amplification.* in *Microtechnologies in Medicine and Biology, 1st Annual International, Conference On. 2000.* 2000.

113. Khandurina, J., T.E. McKnight, S.C. Jacobson, L.C. Waters, R.S. Foote, and J.M. Ramsey, *Integrated system for rapid PCR-based DNA analysis in microfluidic devices.* Analytical Chemistry, 2000. **72**(13): p. 2995-3000.

114. Interfacial-Dynamics, C., *Working With Microspheres.* 2000, Interfacial Dynamics Corporation: Portland,OR, USA.

115. McCarthy, M., *Digital Potentiometers Vary Amplitude in DDS Devices*, in *Electronic Design*. 2000, May 29. p. 108.

116. Whitesides, G.M., E. Ostuni, S. Takayama, X. Jiang, and D.E. Ingber, *Soft lithography in biology and biochemistry.* Annu Rev Biomed Eng, 2001. **3**: p. 335-73.

117. Chou, H.-P., M.A. Unger, and S.R. Quake, *A Microfabricated Rotary Pump.* Biomedical Microdevices, 2000. **3**(4): p. 323-330.

Appendix A

Protocol for creating a solution of BSA coated beads

The following protocol describes the preparation of latex beads to be used in DEP experiments. It has been found that latex beads tend to aggregate, bind and stick to the walls of the channels due to the nature of the PDMS surface, which is hydrophobic, and to the group surface charges on the latex beads, which are also hydrophobic. One way to prevent the aggregation of beads is to sonicate the solution before placing the solution in the channels, and after the experiments has taken place; however, this solution does not prevent the clustering of beads when the experiment is running on. The accumulation of beads is expected due to the polarization of the particles, but they stick together even after the removal of the electric field rendering the cleansing of the chips more laborious and time-consuming.

According to the manufacturer's datasheet (http://www.idclatex.com/body_workingwithmicros.html), a possible solution to this problem is to coat the surface with BSA, egg albuminun, whole serum, or dextrans and maintain a buffer PH > 5. BSA was chosen to be the 'blocking' or coating agent due to its readiness in the lab.

Purpose:

This protocol outlines how Latex beads may be coated with BSA.

Equipment:

20 μL and 200 μL pipettes with several tips
200 μL sample vials
Conductivity meter
PH meter

Reagents:

Latex beads
A stock solution of BSA
Deionized water or Milli-Q ultra pure water

Procedure:

BSA Coated Bead Solution

Add 53 µL of Milli-Q water to a 200 µL sample vial
Add 44 µL of X.X % solids stock solution of Latex beads.

Add 3 µL of 10 mg/mL BSA solution.
Stir and allow to incubate for 5 min. at room temperature.

The solution created should be 1.01% beads by volume and 300µg/µL BSA coated on the beads as suggested by the manufacturer. The calculation are as follows:

Beads:

$$\frac{(X.X\%\_solid\_stock\_solution\_value)(volume\_of\_bead\_solution)}{total\_volume} =$$

$$= (\%\_new\_concentration) \rightarrow \frac{(X.X\%)(44\mu L)}{100\mu L} = 1.01\%\_solid$$

BSA:

$$\frac{(Concentration\_of\_BSA)(Volume\_of\_stock\_used)}{total\_volume} =$$

$$= (Concentration\_of\_BSA) \Rightarrow \frac{\left(10 {}^{\mu g}\!/_{\mu L}\right)(3\mu L)}{100\mu L} = 0.300 {}^{\mu g}\!/_{\mu L}$$

DEP solution

Add 20 mL of Milli-Q water to a labeled 100 mL tube
Completely transfer the BSA coated bead solution to this tube by pipetting the 100 µL volume and then rinsing the vial with another 100 µL aliquot of the fluid in the tube.

The solution in the tube should be 0.04% solids. It is now ready for use, and can be tested for conductivity and PH if desired.

Appendix B

Parts Lists

Pneumatic System

No	Qty	Description	PN	Manufacturer/ Distributor
1	6	23-Gauge Stainless steel Hypodermic Needle L-shape	TBD	New England Small Tube Corp.
2	1	Masterflex® Tygon® tubing, 0.64mm ID, 100 ft/pk, 23 gauge	A-95609-22	Cole-Parmer
3	8	Microtubing, Reducing connector 13 x 20 Gauge	EW-34000-24	Cole-Parmer
4	1	Tygon® LabTubing, 1/16" ID, 1/8" OD, Wall thicknees 1/32"50 ft/pk	A-06408-62	Cole-Parmer
5	6	12V 30 psi High Density Interface (HDI) 3-way solenoid valves, OD 0.080"	LHDA1233115H	The Lee Company
6	2	0 to 30 psi Industrial Gauge, 4 1/2" Dial, 1/4" NPT(M)	EW-68047-12	Cole-Parmer
7	2	66ML Female Connector, NPTF 1/4, Tube size 1/8, Microlok Fitting	66ML-2-4	Parker Pneumatic
8	2	27R Regulator, 1/4" NPT Port, 30 PSIG, Pressure Limiter	27R110AD	Parker Pneumatic
9	2	639PL Plug, 1/4, Prestolok Fitting	639PL-4	Parker Pneumatic
10	2	W68PL Male Connector, NPTF 1/4, Tube size 1/4, Prestolok Fitting	W68PL-4-4	Parker Pneumatic
11	2	639PL Plug, 1/8, Prestolok Fitting	639PL-2	Parker Pneumatic
12	2	W68PL Male Connector, NPTF 1/4, Tube size 1/8, Prestolok Fitting	W68PL-2-4	Parker Pneumatic
13	1	364PL Union Tee, Tube size 1/4	364PL-4	Parker Pneumatic
14	1	Tee Connector with 200 Series Barbs, 1/16" (1.5 mm) ID Tubing PK 100	T210-6	Value Plastics Inc.
15	1	Sterile INTRAMEDIC® Luer-Stub Adapters, 15 gauge, orange	427560	Becton, Dickinson and Company
16	1	Sterile INTRAMEDIC® Luer-Stub Adapters, 17 gauge, grey	427562	Becton, Dickinson and Company
17	1	Sterile INTRAMEDIC® Luer-Stub Adapters, 20 gauge, yellow	427564	Becton, Dickinson and Company
18	1	Sterile INTRAMEDIC® Luer-Stub Adapters, 23 gauge, light green	427565	Becton, Dickinson and Company

Frequency Generator

No	Qty	Description	PN	Distributor/ Manufacturer
1	19	CAP .1UF 25V CERAMIC X7R 1206	PCC1883TR-ND	Digikey (Panasonic)
2	2	CAP CERAMIC 10000PF 50V NP0 1206	PCC2167CT-ND	Digikey (Panasonic)
3	1	IC OP AMP DUAL 8-SOIC	LM1458M-ND	Digikey (National Semiconductor)
4	6	IC CURR-FDBK AMP VIDEO HS 8-SOIC	AD811JR-ND	Digikey (Analog Devices)
5	1	IC DDS DAC 10BIT 50MHZ 16-TSSOP	AD9835BRU-ND	Digikey (Analog Devices)
6	1	IC DGTL POT 8BIT 10K 1CH 8-SOIC	AD8400AR10-ND	Digikey (Analog Devices)
7	2	IC MUX/DEMUX DL ANA 4CH 16-SOIC	CD4052BCM-ND	Digikey (Fairchild Semiconductors)
8	1	Fixed Frequency SMD Oscillators, 50 MHZ 5V 50pF	695-CSX750FCC-50	Mouser(Citizen)
9	2	CONN HEADER 4POS 1.25MM R/A SMD	WM1755-ND	Digikey (Molex)
10	2	CONN HOUSING 4POS 1.25MM	WM1722-ND	Digikey (Molex)
11	8	CONN TERM FEMALE 28-32AWG TIN	WM1775-ND	Digikey (Molex)
12	4	CAP 1.0UF 25V TANT TE SERIES	PCS5105CT-ND	Digikey (Panasonic)
13	1	CAP TANTALUM 100UF 16V 10% SMD	478-1723-1-ND	Digikey (AVX Corp)
14	4	CAP 4.7UF 20V TANTALUM TE SMD	P11322CT-ND	Digikey (Panasonic)
15	1	CAP 10UF 20V TANTALUM TE SMD	P11324CT-ND	Digikey (Panasonic)
16	1	DIODE SCHOTTKY 30V 1A SMB	STPS1L30UIRCT-ND	Digikey (International Rectifier)
17	5	RES 75.0 OHM 1/8W 1% 1206 SMD	P75.0FCT-ND	Digikey (Panasonic)
18	2	RES 750 OHM 1/8W 1% 1206 SMD	P750FCT-ND	Digikey (Panasonic)
19	4	RES 590 OHM 1/8W 1% 1206 SMD	P590FCT-ND	Digikey (Panasonic)
20	1	RES 49.9 OHM 1/8W 1% 1206 SMD	P49.9FCT-ND	Digikey (Panasonic)
21	1	RES 301 OHM 1/8W 1% 1206 SMD	P301FCT-ND	Digikey (Panasonic)
22	3	RES 499 OHM 1/8W 1% 1206 SMD	P499FCT-ND	Digikey (Panasonic)
23	3	RES 1.00K OHM 1/8W 1% 1206 SMD	P1.00KFCT-ND	Digikey (Panasonic)
24	1	RES 10.0K OHM 1/8W 1% 1206 SMD	P10.0KFCT-ND	Digikey (Panasonic)
25	1	RES 100K OHM 1/8W 1% 1206 SMD	P100KFCT-ND	Digikey (Panasonic)
26	4	RES 750K OHM 1/8W 1% 1206 SMD	P750KFCT-ND	Digikey (Panasonic)
27	4	RES 1.00M OHM 1/8W 1% 1206 SMD	P1.00MFCT-ND	Digikey (Panasonic)

Control relays

No	Qty	Description	PN	Distributor/ Manufacturer
1	1	25 PLUG SP/MS CNUT, PCB - Through Hole,	A2102-ND	Digikey (AMP)
2	6	RELAY REED DIP SPST 5V W/DIODE	306-1020-ND	Digikey (Coto Relays)

Appendix C

Electrode and PDMS Masks

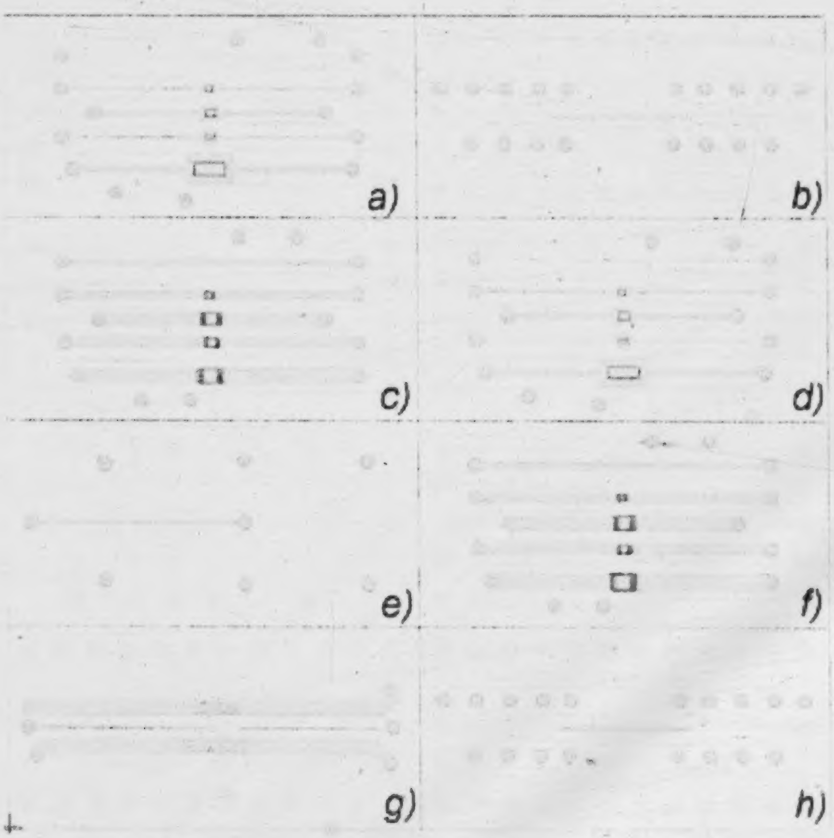


Figure 54. Mask design for electrodes and PDMS channels. (File Name: fingers_castellated.tdb)

Electrode and PDMS masks were created in L-Edit. Five different electrode configurations (interdigitated electrodes, interdigitated castellated electrodes, polynomial electrodes, and series of bus bars for TWD) were designed to fit in a single wafer of 4"x 4".

Figure 54.a (54.d) shows a microfluidic chip design including five interdigitated electrode configurations with gaps of 10, 20, 30, 40 and 50 μm and channel widths of 100 μm, 400 μm, 600 μm, 200 μm, and 800 μm, respectively.

Figure 54.c (54.f) shows the next microfluidic chip design containing five interdigitated castellated electrode configurations with gaps of 20, 40, 80, 60 and 100 µm accompanied with channel widths of 700 µm, 1mm, 1.6 mm, 1.3 mm, and 2.25 mm, respectively.

The right part of Figure 54.e contains a polynomial electrode configuration with a space between tips of 64 µm and a channel width of 30 µm. The left part includes four two-electrode configurations facing each other with gaps of 25, 50, 35 and 50 um and a single channel with a width of 250 µm.

Figure 54.b (54.g) contains an electrode design (prototype) in the form of spiral to be possibly used in travelling wave dielectrophoresis. Figure 54.h (54.b) is a prototype to be used in travelling wave dielectrophoresis.

Appendix D

Printed Circuit Board Layouts

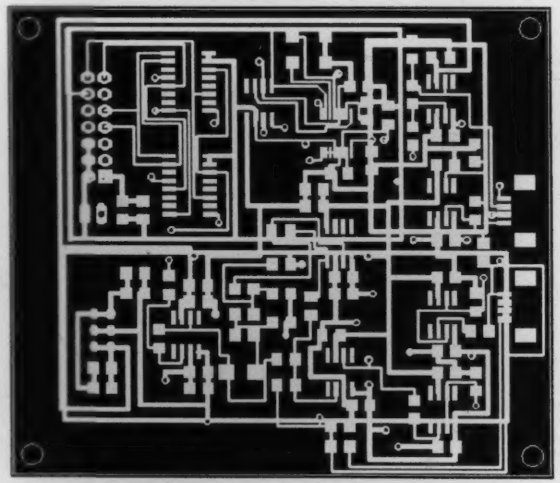


Figure 55. Top layer layout, frequency generator circuit. (File Name: one_freq_gen.brd)

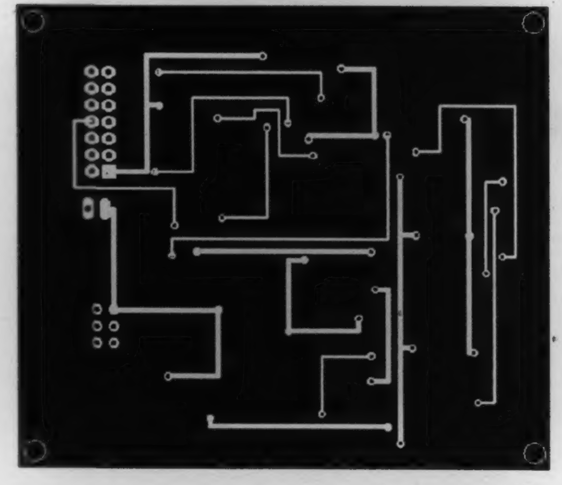


Figure 56. Bottom layer layout, frequency generator circuit. (File Name: one_freq_gen.brd)

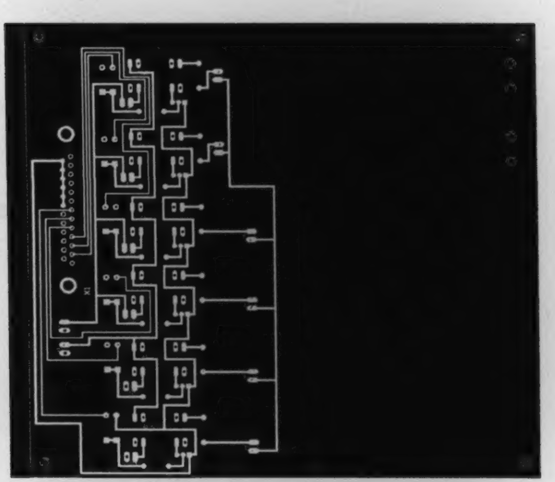


Figure 57. Bottom layer layout for the electronic valve control circuit. (File Name: relays.brd)

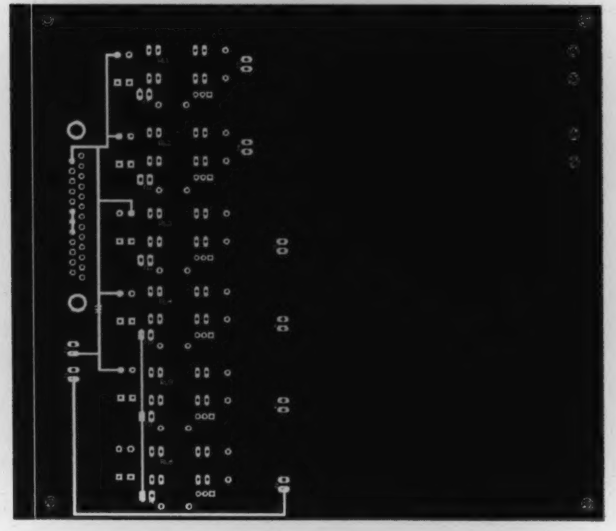


Figure 58. Top layer layout for the electronic valve control circuit. (File Name: relays.brd)

Appendix E

Software Programs: Microcontroller

Software programs for the microcontroller used to program the Digital Direct Synthesizer.

```
#include <16f877.h>
#fuses XT, NOWDT, NOPROTECT, NOPUT, NOLVP,
NOBROWNOUT //configutation bits
#use delay(clock=4000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)// Using
the rs232 as communication with the PIC

#include <stdlib.h>//include standard lib
#include <math.h>//include math for the pow function


#define PIC_CTL_0 PIN_D2   //defining pins with variables
#define PIC_CTL_1 PIN_D3
#define PIC_CTL_2 PIN_D1
#define PIC_CTL_3 PIN_A5

#define PIC_CS PIN_D0
#define PIC_SCLK PIN_C3
#define PIC_FSYNC PIN_C0
#define PIC_SDATA PIN_C5


long getFrequency(void); //prototype for getFrequency
function
float convertFrequency(long);//prototype for
convertFrequency function
void spiTransfer(int32);//prototype for spiTransfer function
void initializeChip(void);//prototpe for initializeChip function
main(void)
{

long frequency; //variables
float data;
int32 hopping;


initializeChip();// call for function initializeChip
printf("\n\r If you wish to change the current states");
printf("\n\r PLease enter frequency in KHZ and ");
printf("\n\r Press q to quit when the frequency entry is set");
printf("\n\r Note that the maximum chip frequency is 50000
KHZ:\n\r");

frequency=getFrequency();//call for function getFrequency
and return value put in frequency
printf("\n\r The frequency entered is %lu",frequency);

data=convertFrequency(frequency);//call to function
convertFrequency and return float in data

printf("\n\r The frequency converted to data sent %f",data);

hopping=(int32)data;//conversion syntax from variable float to
variable int32


spiTransfer(hopping);//call for function spiTrans

printf("\n\rEnd of program !!");

}


long getFrequency(void){

            char s[6];//we expect tno more than 5 character
for 50MHZ and the null character
            long l;

            int full=5;
            char c;
  int i=0;

            do
            {
                        c=getc();
                                    if ((c>'/')&&(c<':'))//the
character inserted should be a number
                        if (i<full)
                                    {
            s[i++]=c;
            putc(c);
                        }


            } while(c!='q'&&i<full);//q is the quitting Key
            s[i]=0;

printf( "\n\r testing for string %S",s);//debugging
l=atol(s);//convert string to long variable
printf("\n\r The frequency entered is %lu",l);
printf("\n\r Inserting frequency completed");
return(l);
}


float convertFrequency(long f){
            float send;
            printf("\n\r in sec func The frequency entered is
%lu",f);
            send=pow(2,32)*f/(50000);//Relation btw the
desired frequency and the sent data
            printf("\n\r The frequency entered is %f",send);
            return send;
}
```

```
void initializeChip(void){
   printf("\n\r Initialization of the DDS Chip");
              printf("\n\r Frequency Reg 0 is set to 1MHZ");
              printf("\n\r Frequency Reg 1 is set to 2MHZ");
              printf("\n\r Phase registers are set to 0 rad");


//Selecting the DDS CHIP
              output_low(PIC_CTL_1);
   output_high(PIC_CTL_2);
              output_low(PIC_CTL_0);
              output_low(PIC_CTL_3);
              delay_us(1);

              //programming the DDS, for a value of 10 MHZ
for register zero
              //50 MHZ for register 1
              // all phase registor are set to zero

              setup_spi(spi_master | spi_h_to_l);



              output_low(PIC_FSYNC);
              spi_write(0xf8);
              spi_write(0x00);
              output_high(PIC_FSYNC);


              // deselect source bit and sync bit . Bits
manipulation and not pins
   output_low(PIC_FSYNC);
              spi_write(0x80);
              spi_write(0x00);
              output_high(PIC_FSYNC);

   // register zero
   //initializing the chip
              output_low(PIC_FSYNC);
              spi_write(0x30);
              spi_write(0x52 );
              output_high(PIC_FSYNC);

              output_low(PIC_FSYNC);
              spi_write(0x21);
              spi_write(0xB8 );
              output_high(PIC_FSYNC);


              output_low(PIC_FSYNC);
              spi_write(0x32);
              spi_write(0x1E);
              output_high(PIC_FSYNC);

              output_low(PIC_FSYNC);
              spi_write(0x23);
              spi_write(0x05);
              output_high(PIC_FSYNC);
              //register 1
              output_low(PIC_FSYNC);
              spi_write(0x34);
              spi_write(0x52);
              output_high(PIC_FSYNC);

              output_low(PIC_FSYNC);
              spi_write(0x25);
              spi_write(0xB8);
              output_high(PIC_FSYNC);
              output_low(PIC_FSYNC);
              spi_write(0x36);
              spi_write(0x1E);
              output_high(PIC_FSYNC);

              output_low(PIC_FSYNC);
              spi_write(0x27);
              spi_write(0x05);
              output_high(PIC_FSYNC);
   //phase register 0
              output_low(PIC_FSYNC);
              spi_write(0x18);
              spi_write(0x00);
              output_high(PIC_FSYNC);

   output_low(PIC_FSYNC);
              spi_write(0x09);
              spi_write(0x00);
              output_high(PIC_FSYNC);
   //PREG 1
              output_low(PIC_FSYNC);
              spi_write(0x1A);
              spi_write(0x00);
              output_high(PIC_FSYNC);

   output_low(PIC_FSYNC);
              spi_write(0x0B);
              spi_write(0x00);
              output_high(PIC_FSYNC);
   //PREG 2
              output_low(PIC_FSYNC);
              spi_write(0x1c);
              spi_write(0x00);
              output_high(PIC_FSYNC);

              output_low(PIC_FSYNC);
              spi_write(0x0d);
              spi_write(0x00);
              output_high(PIC_FSYNC);
              //PREG 3
              output_low(PIC_FSYNC);
              spi_write(0x1e);
              spi_write(0x00);
              output_high(PIC_FSYNC);

   output_low(PIC_FSYNC);
              spi_write(0x0f);
              spi_write(0x00);
              output_high(PIC_FSYNC);

   // display
              output_low(PIC_FSYNC);
              spi_write(0xc0);
              spi_write(0x00);
              output_high(PIC_FSYNC);

}

void spiTransfer(int32 d){



              int low1;
              int low2;
              int high1;
              int high2;
              short val1;
              short val0;
              short value;
```

```
int counter0;
            int counter1;
int counter2;
int counter3;
            int counter4;
            int newcount2;
            int newcount3;
            int newcount4;
 int32 oper;

            oper=171798665;//debugging variable



            val1=1;
            val0=0;


            printf("\n\r Testing with oper int32 variable");

            for (counter0=0;counter0<32;counter0++){
                        if(bit_test(oper,counter0)){
                                    printf(" %U",val1);}
            if(!bit_test(oper,counter0)){
                        printf(" %U",val0);}
            }


            printf("\n\r the int32 value in the trans %lu",d);
            printf("\n\r least significant bit printed first \n\r");



            for (counter0=0;counter0<32;counter0++){
                        if(bit_test(d,counter0)){
                                    printf(" %U",val1);}
            if(!bit_test(d,counter0)){
                        printf(" %U",val0);}
            }


            // The codes below are the one effecting the data
sent to the Analog device

            //extracting the low 8 bit value
            for (counter1=0;counter1<8;counter1++){
                        value=bit_test(d,counter1);
                        if(value==1)
                                    bit_set(low1,counter1);
                        if(value==0)
                                    bit_clear(low1,counter1);
            }
            printf("\n\r The low1 byte is in decimal %U",low1);

 // extracting the next 8 bit value
            for (counter2=8;counter2<16;counter2++){

                        newcount2=counter2-8;
                        value=bit_test(d,counter2);
                        if(value==1)

            bit_set(low2,newcount2);
                        if(value==0)

            bit_clear(low2,newcount2);
            }
            printf("\n\r The low2 byte is in decimal %U",low2);

// extracting the high 8 bit value
            for (counter3=16;counter3<24;counter3++){
```

```
                        newcount3=counter3-16;
                        value=bit_test(d,counter3);
                        if(value==1)

            bit_set(high1,newcount3);
                        if(value==0)

            bit_clear(high1,newcount3);
            }
            printf("\n\r The high1 byte is in decimal
%U",high1);


            //extracting the highest 8 bit value
            for (counter4=24;counter4<32;counter4++){

                        newcount4=counter4-24;
                        value=bit_test(d,counter4);
                        if(value==1)

            bit_set(high2,newcount4);
                        if(value==0)

            bit_clear(high2,newcount4);
            }
            printf("\n\r The high2 byte is in decimal
%U",high2);




            //selecting DDS chip
            output_low(PIC_CTL_1);
            output_high(PIC_CTL_2);
            output_low(PIC_CTL_0);
            output_low(PIC_CTL_3);
            delay_us(1);
  output_high(PIC_FSYNC);

            setup_spi(spi_master | spi_h_to_l);

  // freq register zero
            //sending the required bits
            output_low(PIC_FSYNC);
            spi_write(0x30);
            spi_write(low1);
            output_high(PIC_FSYNC);

            output_low(PIC_FSYNC);
            spi_write(0x21);
            spi_write(low2 );
            output_high(PIC_FSYNC);


            output_low(PIC_FSYNC);
            spi_write(0x32);
            spi_write(high1);
            output_high(PIC_FSYNC);

            output_low(PIC_FSYNC);
            spi_write(0x23);
            spi_write(high2);
            output_high(PIC_FSYNC);

            output_low(PIC_FSYNC);
            spi_write(0xb0);
            spi_write(0x00);
            output_high(PIC_FSYNC);
```

```
output_low(PIC_FSYNC);
spi_write(0x60);
spi_write(0x00);
output_high(PIC_FSYNC);
```

```
        printf("\n\r The SPI Transfer is completed");
}
```

Appendix F

Software Tools

Example of text file created by the program

```
Name of the file: Chip Spiral 1 Latex Beads 2003 10 24 h5m13.dep

Univesity of Alberta - Backhouse Lab
Experiment took place on:
year 2003, month 10, day 24 hour 5 : 13
Dielectrophoresis setup
****************************

Chip Information
****************************

Spiral_1
****************************

Cell Information
****************************

Type of Cells: Latex Beads
Conductivity: 4.0000 S/m
Cell Size: 10.00 um
Cell Permittivity: 3.00 E/Eo
Cell Surface Groups: NH2
Cells delivered by: Jose Garcia
******************************

Medium Information
******************************

Medium Reagents: Water-Salt
Conductivity: 3.0000 S/m
Humidity: 50.0000
PH: 4.0000
Temp: 24.0000 C
Medium Prepared on: Backhouse Lab
Prepared By:
******************************

General Comments
******************************

Showed good performance
******************************

Comments on the chip
******************************

Washed with IPH before testings.
```

Source Code

File Name: dep_controllerDlg.cpp

```
// dep_controllerDlg.cpp : implementation file
//

#include "stdafx.h"
#include "dep_controller.h"
#include "dep_controllerDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////// Constants/////////////////////////////////
const float X_EDGE_BUFFER=0.05f;            // Pixel buffer
between bitmap and window edge
const float Y_EDGE_BUFFER=0.05f;            // (represented in
composition space [0 - 1.0f])
#define WM_GRAPHNOTIFY WM_APP + 1         // Private
Message used by event handler
typedef LPBITMAPINFOHEADER PDIB;            //Constants for stll
images
#define BFT_BITMAP 0x4d42                              // 'BM'

////////// Variables ///////////////////

////////// Macros ///////////////////////////////////
#define DibNumColors(lpbi)      ((lpbi)->biClrUsed == 0 && (lpbi)-
>biBitCount <= 8 \
                              ? (int)(1 << (int)(lpbi)->biBitCount)       \
                              : (int)(lpbi)->biClrUsed)

#define DibSize(lpbi)         ((lpbi)->biSize + (lpbi)->biSizeImage +
(int)(lpbi)->biClrUsed * sizeof(RGBQUAD))

#define DibPaletteSize(lpbi)    (DibNumColors(lpbi) *
sizeof(RGBQUAD))


/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
        CAboutDlg();

// Dialog Data
        //{{AFX_DATA(CAboutDlg)
        enum { IDD = IDD_ABOUTBOX };
        //}}AFX_DATA

        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CAboutDlg)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);
// DDX/DDV support
        //}}AFX_VIRTUAL

// Implementation
protected:
        //{{AFX_MSG(CAboutDlg)
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
        //{{AFX_DATA_INIT(CAboutDlg)
        //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CAboutDlg)
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
        //{{AFX_MSG_MAP(CAboutDlg)
                // No message handlers
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CDep_controllerDlg dialog

CDep_controllerDlg::CDep_controllerDlg(CWnd* pParent
/*=NULL*/)
        : CDialog(CDep_controllerDlg::IDD, pParent)
{
        //{{AFX_DATA_INIT(CDep_controllerDlg)
        m_fCellCond = 0.0f;
        m_fCellPerm = 0.0f;
        m_sCellPrep = _T("");
        m_fCellSize = 0.0f;
        m_sCellSurf = _T("");
        m_sCellType = _T("");
        m_sChipCode = _T("");
        m_sCommentsChip = _T("");
        m_sCommentsGral = _T("");
        m_fMediumCond = 0.0f;
        m_sMediumPrep = _T("");
        m_sMediumReagent = _T("");
        m_sStatus = _T("");
        m_sStatus2 = _T("");
        m_fAutFreqMax = 0.0f;
        m_fAutFreqMin = 0.0f;
        m_fAutVoltInt = 0.0f;
        m_fAutVoltMax = 0.0f;
        m_fAutVoltMin = 0.0f;
        m_fGenOffset = 0.0f;
        m_fManFreq = 0.0f;
        m_fManVolt = 0.0f;
        m_fMediumHumd = 0.0f;
        m_fMediumPH = 0.0f;
        m_fMediumTemp = 0.0f;
        m_fAutFreqInt = 0.0f;
        m_dAutTimerInt = 0;
        //}}AFX_DATA_INIT
        // Note that LoadIcon does not require a subsequent
DestroyIcon in Win32
        m_hIcon = AfxGetApp()-
>LoadIcon(IDR_MAINFRAME);
}

void CDep_controllerDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CDep_controllerDlg)
        DDX_Control(pDX, IDC_AUT_STOP, m_ctlAutStop);
        DDX_Control(pDX, IDC_MAN_FREQ_SEL,
m_cmbManFreqSel);
        DDX_Control(pDX, IDC_GEN_WAVEFORM,
m_cmbGenWaveform);
        DDX_Control(pDX, IDC_GEN_TERMINATION,
m_cmbGenTermination);
        DDX_Control(pDX, IDC_AUT_FREQ_SEL_MIN,
m_cmbAutFreqSelMin);
        DDX_Control(pDX, IDC_AUT_FREQ_SEL_MAX,
m_cmbAutFreqSelMax);
        DDX_Control(pDX, IDC_AUT_FREQ_SEL_INT,
m_cmbAutFreqSelInt);
        DDX_Text(pDX, IDC_CELL_CONDUCTIVITY,
m_fCellCond);
        DDX_Text(pDX, IDC_CELL_PERMITTIVITY,
m_fCellPerm);
        DDX_Text(pDX, IDC_CELL_PREPARATION,
m_sCellPrep);
        DDX_Text(pDX, IDC_CELL_SIZE, m_fCellSize);
        DDX_Text(pDX, IDC_CELL_SURFACE, m_sCellSurf);
        DDX_Text(pDX, IDC_CELL_TYPE, m_sCellType);
        DDX_Text(pDX, IDC_CHIP_CODE, m_sChipCode);
        DDX_Text(pDX, IDC_COMMENTS_CHIP,
m_sCommentsChip);
        DDX_Text(pDX, IDC_COMMENTS_GRAL,
m_sCommentsGral);
        DDX_Text(pDX, IDC_MEDIUM_CONDUCTIVITY,
m_fMediumCond);
        DDX_Text(pDX, IDC_MEDIUM_PREPARATION,
m_sMediumPrep);
        DDX_Text(pDX, IDC_MEDIUM_REAGENTS,
m_sMediumReagent);
        DDX_Text(pDX, IDC_STATUS, m_sStatus);
        DDX_Text(pDX, IDC_STATUS2, m_sStatus2);
        DDX_Text(pDX, IDC_AUT_FREQ_MAX,
m_fAutFreqMax);
        DDX_Text(pDX, IDC_AUT_FREQ_MIN,
m_fAutFreqMin);
        DDX_Text(pDX, IDC_AUT_VOLT_INT, m_fAutVoltInt);
        DDX_Text(pDX, IDC_AUT_VOLT_MAX,
m_fAutVoltMax);
        DDX_Text(pDX, IDC_AUT_VOLT_MIN,
m_fAutVoltMin);
        DDX_Text(pDX, IDC_GEN_OFFSET, m_fGenOffset);
        DDX_Text(pDX, IDC_MAN_FREQ, m_fManFreq);
        DDX_Text(pDX, IDC_MAN_VOLT, m_fManVolt);
        DDX_Text(pDX, IDC_MEDIUM_HUMID,
m_fMediumHumd);
        DDX_Text(pDX, IDC_MEDIUM_PH, m_fMediumPH);
        DDX_Text(pDX, IDC_MEDIUM_TEMP,
m_fMediumTemp);
        DDX_Text(pDX, IDC_AUT_FREQ_INT,
m_fAutFreqInt);
        DDX_Text(pDX, IDC_AUT_TIME_INT,
m_dAutTimerInt);
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CDep_controllerDlg, CDialog)
        //{{AFX_MSG_MAP(CDep_controllerDlg)
        ON_WM_SYSCOMMAND()
        ON_WM_PAINT()
        ON_WM_QUERYDRAGICON()
        ON_BN_CLICKED(IDC_STORE_DATA, OnStoreData)
        ON_BN_CLICKED(IDC_CLEAR_DATA, OnClearData)
        ON_BN_CLICKED(IDC_STORE_COMMENTS,
OnStoreComments)
        ON_BN_CLICKED(IDC_CLEAR_COMMENTS,
OnClearComments)
        ON_BN_CLICKED(IDC_NEW_DATA, OnNewData)
        ON_BN_CLICKED(IDC_AUT_APPLY, OnAutApply)
        ON_BN_CLICKED(IDC_AUT_STOP, OnAutStop)
        ON_BN_CLICKED(IDC_AUT_RES, OnAutRes)
        ON_BN_CLICKED(IDC_MAN_APPLY, OnManApply)
        ON_WM_TIMER()
        ON_BN_CLICKED(IDC_STOPCAPTURE,
OnStopcapture)
        ON_BN_CLICKED(IDC_PLAYVIDEO, OnPlayvideo)
        ON_BN_CLICKED(IDC_STILL, OnStill)
        ON_BN_CLICKED(IDC_CAPTURE, OnCapture)
        ON_BN_CLICKED(IDC_RECORD, OnRecord)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CDep_controllerDlg message handlers

BOOL CDep_controllerDlg::OnInitDialog()
{
        CDialog::OnInitDialog();

        // Add "About..." menu item to system menu.

        // IDM_ABOUTBOX must be in the system command
range.
        ASSERT((IDM_ABOUTBOX & 0xFFF0) ==
IDM_ABOUTBOX);
        ASSERT(IDM_ABOUTBOX < 0xF000);

        CMenu* pSysMenu = GetSystemMenu(FALSE);
        if (pSysMenu != NULL)
        {
                CString strAboutMenu;

        strAboutMenu.LoadString(IDS_ABOUTBOX);
                if (!strAboutMenu.IsEmpty())
                {
                        pSysMenu-
>AppendMenu(MF_SEPARATOR);
                        pSysMenu-
>AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
                }
        }

        // Set the icon for this dialog.  The framework does this
automatically
        //  when the application's main window is not a dialog
        SetIcon(m_hIcon, TRUE);
        // Set big icon
        SetIcon(m_hIcon, FALSE);                        // Set
small icon

        // TODO: Add extra initialization here

        //Set default values
        m_cmbAutFreqSelInt.SetCurSel(0);
        m_cmbAutFreqSelMax.SetCurSel(0);
        m_cmbAutFreqSelMin.SetCurSel(0);
        m_cmbGenTermination.SetCurSel(0);
        m_cmbGenWaveform.SetCurSel(0);
        m_cmbManFreqSel.SetCurSel(0);

        //Initialize Serial Port
        CDep_controllerDlg::OpenCommPort();

        //Allocate memory or release?
        ZeroMemory(&bmpInfo, sizeof(bmpInfo) );

        //Clean Variables
        g_fVoltOut = 0;
```

```
            g_fFreqOut = 0;
            g_dFlagTimer = 0;
            g_dPictureCounter = 0;

            return TRUE;  // return TRUE  unless you set the focus
to a control
}

void CDep_controllerDlg::OnSysCommand(UINT nID, LPARAM
lParam)
{
            if ((nID & 0xFFF0) == IDM_ABOUTBOX)
            {
                        CAboutDlg dlgAbout;
                        dlgAbout.DoModal();
            }
            else
            {
                        CDialog::OnSysCommand(nID, lParam);
            }
}

// If you add a minimize button to your dialog, you will need the code
below
//  to draw the icon.  For MFC applications using the document/view
model,
//  this is automatically done for you by the framework.

void CDep_controllerDlg::OnPaint()
{
            if (IsIconic())
            {
                        CPaintDC dc(this); // device context for
painting

                        SendMessage(WM_ICONERASEBKGND,
(WPARAM) dc.GetSafeHdc(), 0);

                        // Center icon in client rectangle
                        int cxIcon =
GetSystemMetrics(SM_CXICON);
                        int cyIcon =
GetSystemMetrics(SM_CYICON);
                        CRect rect;
                        GetClientRect(&rect);
                        int x = (rect.Width() - cxIcon + 1) / 2;
                        int y = (rect.Height() - cyIcon + 1) / 2;

                        // Draw the icon
                        dc.DrawIcon(x, y, m_hIcon);
            }
            else
            {
                        CDialog::OnPaint();
            }
}

// The system calls this to obtain the cursor to display while the user
drags
//  the minimized window.
HCURSOR CDep_controllerDlg::OnQueryDragIcon()
{
            return (HCURSOR) m_hIcon;
}

void CDep_controllerDlg::OnStoreData()
{
            // TODO: Add your control notification handler code
here
            /*****************************************************************/
            // Name : Jose Garcia - University of Alberta -
Backhouse Lab
            // Created on October 18th, 2003
            // Description :
            //                        Algorithm associated with the
Store Parameter Button.
            //                        Captures data entered by the
user and stores it in a file.
            //
            // Modified by :
            // Comments :
            /*****************************************************************/

            // Update all message variables with the data entered
by the user
            UpdateData(TRUE);


            ////////////////////
            //Create File
            CDep_controllerDlg::Create_File();
            ////////////////////

            //////////////////
            //Write Data to File
            //////////////////

            //Declaration of Variables
            CString sHeader, sDataToWrite, sTempBuffer, flag;
//DELETE FLAG AT THE END
            DWORD dStringLength = 0, dBytesWritten;

            //////////////////
            //Header
            //////////////////

            sHeader =  "Univesity of Alberta - Backhouse Lab\n";
            sHeader += "Experiment took place on: \n\t" +
m_sStatus2 + "\n";
            sHeader += "Dielectrophoresis setup\n\n"; //MODIFY
LATER

            //////////////////
            //Chip Information
            //////////////////
            sHeader += "****************************\n";
            sHeader += "\tChip Information\n";
            sHeader += "****************************\n";
            sHeader += m_sChipCode + "\n\n";
            //////////////////
            //Cell Parameters
            //////////////////
            sHeader += "****************************\n";
            sHeader += "\tCell Information\n";
            sHeader += "****************************\n";
            sDataToWrite = sHeader;
            sDataToWrite += CString("Type of Cells: ") +
m_sCellType + "\n";
            sTempBuffer.Format("Conductivity: %.4f S/m\n",
m_fCellCond);
            sDataToWrite += sTempBuffer;
            sTempBuffer.Format("Cell Size: %.2f um\n",
m_fCellSize);
            sDataToWrite += sTempBuffer;
            sTempBuffer.Format("Cell Permittivity: %.2f E/Eo\n",
m_fCellPerm);
            sDataToWrite += sTempBuffer;
            sDataToWrite += CString("Cell Surface Groups: ") +
m_sCellSurf + "\n";
```

```
            sDataToWrite += CString("Cells delivered by: ") +
m_sCellPrep + "\n\n";

            dStringLength = strlen(sDataToWrite);

            ////////////////////
            //Just for the Record
            flag.Format(" length %d ", dStringLength);
            m_sStatus += sDataToWrite + flag;
            ////////////////////

            if (WriteFile(hdlFile, sDataToWrite, dStringLength,
&dBytesWritten, NULL) == TRUE)
                        m_sStatus2 = "Writing to File";
            else
                        m_sStatus2 = "Error in writing to file";

            ////////////////////
            //Medium Parameters
            ////////////////////
            sHeader = "******************************\n";
            sHeader += "\tMedium Information\n";
            sHeader += "******************************\n";
            sDataToWrite = sHeader;
            sDataToWrite += CString("Medium Reagents: ") +
m_sMediumReagent + "\n";
            sTempBuffer.Format("Conductivity: %.4f uS/cm\n",
m_fMediumCond);
            sDataToWrite += sTempBuffer;
            sTempBuffer.Format("Humidity:
%.4f\n",m_fMediumHumd);
            sDataToWrite += sTempBuffer;
            sTempBuffer.Format("PH: %.4f\n",m_fMediumPH);
            sDataToWrite += sTempBuffer;
            sTempBuffer.Format("Temp: %.4f
C\n",m_fMediumTemp);
            sDataToWrite += sTempBuffer;
            sDataToWrite += CString("Medium Prepared on: ") +
m_sMediumPrep + "\n";
            sDataToWrite += CString("Prepared By: ") +
/*m_sCellPrep +*/ "\n\n";
            dStringLength = strlen(sDataToWrite);

            ////////////////////
            //Just for the Record
            flag.Format(" length %d ", dStringLength);
            m_sStatus += sDataToWrite + flag;
            ////////////////////

            if (WriteFile(hdlFile, sDataToWrite, dStringLength,
&dBytesWritten, NULL) == TRUE)
                        m_sStatus2 = "Writing to File";
            else
                        m_sStatus2 = "Error in writing to file";


            //CloseHandle(hdlFile);

            UpdateData(FALSE);
}

void CDep_controllerDlg::OnClearData()
{
            // TODO: Add your control notification handler code
here
            /***************************************************************/
            // Name : Jose Garcia - University of Alberta -
Backhouse Lab
            // Created on October 18th, 2003
            // Description :
            //                          Algorithm associated with the
Clear Data Button.
            //                          Resets all the data contained
in the edit boxes.
            //
            // Modified by :
            // Comments :
            /***************************************************************/

            m_sCellPrep = "";
            m_sCellSurf = "";
            m_sCellType = "";
            m_sChipCode = "";
            m_sMediumPrep = "";
            m_sMediumReagent = "";
            m_fCellCond = 0;
            m_fCellPerm = 0;
            m_fCellSize = 0;
            m_fMediumCond = 0;

            UpdateData(FALSE);
}

void CDep_controllerDlg::OnStoreComments()
{
            // TODO: Add your control notification handler code
here

            /***************************************************************/
            // Name : Jose Garcia - University of Alberta -
Backhouse Lab
            // Created on October 18th, 2003
            // Description :
            //                          Algorithm associated with the
Store Parameter Button.
            //                          Captures data entered by the
user and stores it in a file.
            //
            // Modified by :
            // Comments :
            /***************************************************************/

            UpdateData(TRUE);
            CString sHeader, sDataToWrite, flag; //DELETE FLAG
AT THE END
            DWORD dStringLength = 0, dBytesWritten;
            ////////////////////
            //General Comments
            ////////////////////
            sHeader = "******************************\n";
            sHeader += "\tGeneral Comments\n";
            sHeader += "******************************\n";
            sDataToWrite = sHeader;
            sDataToWrite += m_sCommentsGral + "\n\n";

            ////////////////////
            //Comments On Chip
            ////////////////////
            sHeader = "******************************\n";
            sHeader += "\tComments on the chip\n";
            sHeader += "******************************\n";
            sDataToWrite += sHeader;
            sDataToWrite += m_sCommentsChip + "\n\n";
            dStringLength = strlen(sDataToWrite);

            ////////////////////
            //Just for the Record
            flag.Format(" length %d ", dStringLength);
            m_sStatus += sDataToWrite + flag;
            ////////////////////
```

```
		if (WriteFile(hdlFile, sDataToWrite, dStringLength,
&dBytesWritten, NULL) == TRUE)
				m_sStatus2 = "Writing to File On Store
Comments";
		else
				m_sStatus2 = "Error in writing to file On
Store Comments";

}

void CDep_controllerDlg::OnCancel()
{
		// TODO: Add extra cleanup here

		//Check if a file exists!!!!
		CloseHandle(hdlFile);
		//Check any error here
		//Close Handle for Comm port
		if(CloseHandle(hComm) == FALSE)
				m_sStatus += "Error closing handle\n";
		else
				m_sStatus += "Closing COMM1 port \n";

		OnOK();
}

void CDep_controllerDlg::OnClearComments()
{
		// TODO: Add your control notification handler code
here
		m_sCommentsChip = "";
		m_sCommentsGral = "";
		UpdateData(FALSE);
}

void CDep_controllerDlg::OnNewData()
{
		// TODO: Add your control notification handler code
here
		m_sCellPrep = "";
		m_sCellSurf = "";
		m_sCellType = "";
		m_sChipCode = "";
		m_sMediumPrep = "";
		m_sMediumReagent = "";
		m_fCellCond = 0;
		m_fCellPerm = 0;
		m_fCellSize = 0;
		m_fMediumCond = 0;
		m_sCommentsChip = "";
		m_sCommentsGral = "";
		UpdateData(FALSE);
		CloseHandle(hdlFile);
}

void CDep_controllerDlg::Create_File()
{
		/*************************************************************/
		// Name : Jose Garcia - University of Alberta -
Backhouse Lab
		// Created on October 18th, 2003
		// Description :
		//				Creates a New File
		//
		//
		// Modified by :
		// Comments :
		/*************************************************************/

		//Retrieve local date and time
		SYSTEMTIME sysCurrentTime;
		GetLocalTime(&sysCurrentTime);

		//Display information on screen
		m_sStatus2.Format("year %d, month %d, day %d hour
%d : %d",

		sysCurrentTime.wYear, sysCurrentTime.wMonth,
sysCurrentTime.wDay,

		sysCurrentTime.wHour, sysCurrentTime.wMinute);

		//setFileName including Chip No, Type of Cell, Date,
Time,
		CString tmpFileName, tmpFileTime,
tmpFileNameVideo, FileNameText;
		//Define Path
		tmpFileName = "c:\\thesis\\software\\data\\";

		tmpFileTime.Format("_%d_%d_%d_h%dm%d",

		sysCurrentTime.wYear, sysCurrentTime.wMonth,
sysCurrentTime.wDay,

		sysCurrentTime.wHour, sysCurrentTime.wMinute);

		sFileName = tmpFileName + CString("Chip_") +
m_sChipCode + "_" + m_sCellType + tmpFileTime;


		FileNameVideo =		sFileName + ".avi";

		FileNameText =		sFileName + ".dep";


		//File declarations and configuration
		//HANDLE hdlFile; //make it global //A - now is a global
variable
		LPCSTR FileName = FileNameText;
		//LPCSTR FileName =
"c:\\thesis\\software\\data\\texto1.dep";

		////////////////
		//Create File
		////////////////

		m_sStatus = "Creating File"; UpdateData(FALSE);
//Display information on screen
		hdlFile = CreateFile(FileName, GENERIC_WRITE, 0,
NULL, CREATE_NEW,

		FILE_ATTRIBUTE_NORMAL, NULL);

		if(hdlFile == INVALID_HANDLE_VALUE)
				m_sStatus.Format( "Error in Creating File
%d", GetLastError());
		else
				m_sStatus = "Succeded";

}

void CDep_controllerDlg::OnAutApply()
{
		// TODO: Add your control notification handler code
here

		UpdateData(TRUE);
		CString sTemp;
		g_dFlagTimer = 0;
		//float fAutFreqMax, fAutFreqMin, fAutFreqInt;
		//float fAutFreqInt;
		g_fAutFreqInt = m_fAutFreqInt;
		g_fAutFreqMin = m_fAutFreqMin;
		g_fAutFreqMax = m_fAutFreqMax;
		g_fAutVoltMin = m_fAutVoltMin;
		g_fAutVoltMax = m_fAutVoltMax;
		g_fAutVoltInt = m_fAutVoltInt;

		//CDep_controllerDlg::SwapRealVal(&fAutFreqMax,
m_cmbAutFreqSelMax.GetCurSel());
		//CDep_controllerDlg::SwapRealVal(&fAutFreqMin,
m_cmbAutFreqSelMin.GetCurSel());
		//CDep_controllerDlg::SwapRealVal(&fAutFreqInt,
m_cmbAutFreqSelInt.GetCurSel());

		//frequencies
		CDep_controllerDlg::SwapRealVal(&g_fAutFreqMax,
m_cmbAutFreqSelMax.GetCurSel());
		CDep_controllerDlg::SwapRealVal(&g_fAutFreqMin,
m_cmbAutFreqSelMin.GetCurSel());
		CDep_controllerDlg::SwapRealVal(&g_fAutFreqInt,
m_cmbAutFreqSelInt.GetCurSel());

		//Initial voltage
		g_fVoltOut = g_fAutVoltMin;
		g_fFreqOut = g_fAutFreqMin;
		//BlendApplication again to display information
		CDep_controllerDlg::BlendApplicationAgain();

		sTemp.Format("FREQ %.2f\n",g_fFreqOut);
		sDataCommOut = (LPCSTR) sTemp;
		CDep_controllerDlg::SendDataOutComm();

		sTemp.Format("VOLT %.2f\n",g_fVoltOut);
		sDataCommOut = (LPCSTR) sTemp;
		CDep_controllerDlg::SendDataOutComm();

		//////////////////////////////////////////////////


		gdTotalAutVoltInt = (int) (g_fAutVoltMax -
g_fAutVoltMin)/ g_fAutVoltInt;
		// total number of intervals
		gdTotalAutFreqInt = (int) (g_fAutFreqMax -
g_fAutFreqMin)/ g_fAutFreqInt;

		gdCounterFreq = 0; //initiate to 0
		gdCounterVolt = 0; //initiate to 0

		//For the record
		sTemp.Format("__VOLT TOT %d FREQ TOT
%d____\n", gdTotalAutVoltInt, gdTotalAutFreqInt);
		m_sStatus2 += sTemp;

		// Set time interval interruptions
		SetTimer(ID_TIMER_AUT, m_dAutTimerInt, NULL);

		//Use Signal to output values to the Frequency
Generator

		UpdateData(FALSE);
}

void CDep_controllerDlg::OnAutStop()
{
		// TODO: Add your control notification handler code
here
		KillTimer(ID_TIMER_AUT);
}

void CDep_controllerDlg::SwapRealVal(float *Value, int Selection)
{
		switch(Selection)
		{
		case 0:
				break;
		case 1:
				(*Value) *= 1000; //1KHz
				break;
		case 2:
				(*Value) *= 1000000; //MHz
				break;
		}

}

void CDep_controllerDlg::OnAutRes()
{
		// TODO: Add your control notification handler code
here
		m_fAutFreqInt = 0;
		m_fAutFreqMax = 0;
		m_fAutFreqMin = 0;
		m_dAutTimerInt = 0;
		m_fAutVoltInt = 0;
		m_fAutVoltMax = 0;
		m_fAutVoltMin = 0;
		UpdateData(FALSE);
}

void CDep_controllerDlg::OnManApply()
{
		// TODO: Add your control notification handler code
here
		UpdateData(TRUE);
		float fManFreq;
		CString sTemp;
		fManFreq = m_fManFreq;
		CDep_controllerDlg::SwapRealVal(&fManFreq,
m_cmbManFreqSel.GetCurSel());
		sTemp.Format("frequency : %.2fHz Voltage: %.2f",
fManFreq, m_fManVolt);
		m_sStatus2 += sTemp + "\n";
		//Send Out Comm Voltage
		g_fFreqOut = fManFreq;
		g_fVoltOut = m_fManVolt;
		sTemp.Format("VOLT %.2f\n", g_fVoltOut);
		sDataCommOut = (LPCSTR) sTemp;
		CDep_controllerDlg::SendDataOutComm();
		sTemp.Format("FREQ %.2f\n",g_fFreqOut);
		sDataCommOut = (LPCSTR) sTemp;
		CDep_controllerDlg::SendDataOutComm();
		//Blend Image

		CDep_controllerDlg::BlendApplicationAgain();
		UpdateData(FALSE);
}

HRESULT CDep_controllerDlg::BlendApplicationImage()
{


		HRESULT hr;
		//Get Video Size
		CDep_controllerDlg::BlendApplicationGetVideoSize();
	// Create a device context compatible with the current window
		CClientDC hdc (this);
	HFONT hOldFont = (HFONT) SelectObject(hdc, hFont);
```

```
  // Determine the length of the string, then determine the
  // dimensions (in pixels) of the character string using the
  // currently selected font.  These dimensions are used to create
  // a bitmap below.
  int nTextBmpWidth, nTextBmpHeight;

            CString sDataTemp, sTest;
            sDataTemp.Format(" %.2f V %.2f Hz", g_fVoltOut,
g_fFreqOut);

            // m_sChipCode + "V" "Hz"
            //sTest = "5 MHz 5 Volt DC chip1";-
            sTest = "Chip " + m_sChipCode + sDataTemp;
            CSize zDimm;
            zDimm = hdc.GetTextExtent(sTest);
   nTextBmpHeight = zDimm.cy;
   nTextBmpWidth  = zDimm.cx;

   // Create a new bitmap that is compatible with the current window
            HBITMAP hbm = CreateCompatibleBitmap(hdc,
nTextBmpWidth, nTextBmpHeight);

   // Select our bitmap into the device context and save the old one
   BITMAP bm;
   HBITMAP hbmOld;
   GetObject(hbm, sizeof(bm), &bm);
   hbmOld = (HBITMAP)SelectObject(hdc, hbm);

   // Set initial bitmap settings
   RECT rcText;
   SetRect(&rcText, 0, 0, nTextBmpWidth, nTextBmpHeight);
   SetBkColor(hdc, RGB(255, 255, 255));             // Pure white
background
   SetTextColor(hdc, RGB(255, 0, 0));     // Write text with
requested color RED

            // Draw the requested text string onto the bitmap
            hdc.TextOut(0,0, sTest);


   // Configure the VMR's bitmap structure
   //VMR9AlphaBitmap bmpInfo; //made global
   //ZeroMemory(&bmpInfo, sizeof(bmpInfo) );
   bmpInfo.dwFlags = VMR9AlphaBitmap_hDC;
   bmpInfo.hdc = hdc;
                              // DC which has selected our bitmap

   // Remember the width of this new bitmap
   dImageWidth = bm.bmWidth;

   // Save the ratio of the bitmap's width to the width of the video file.
   // This value is used to reposition the bitmap in composition
space.
   fBitmapCompWidth = (float)dImageWidth / (float)g_lVideoWidth;

   // Display the bitmap in the bottom right corner.
   // rSrc specifies the source rectangle in the GDI device context
   // rDest specifies the destination rectangle in composition space
(0.0f to 1.0f)
   bmpInfo.rDest.left  = 0.0f + X_EDGE_BUFFER;
   bmpInfo.rDest.right = 1.0f - X_EDGE_BUFFER;
   bmpInfo.rDest.top = (float)(g_lVideoHeight - bm.bmHeight) /
(float)g_lVideoHeight - Y_EDGE_BUFFER;
   bmpInfo.rDest.bottom = 1.0f - Y_EDGE_BUFFER;


   // Transparency value 1.0 is opaque, 0.0 is transparent.
   //bmpInfo.fAlpha = TRANSPARENCY_VALUE;

            bmpInfo.fAlpha = 0.8f;

            // Set the COLORREF so that the bitmap outline will be
transparent
   bmpInfo.clrSrcKey = RGB(255, 255, 255);  // Pure white
   bmpInfo.dwFlags |= VMRBITMAP_SRCCOLORKEY;

            // put text on screen
            bmpInfo.rSrc = rcText;

            //Blend Image second step, calling Query Interface and
IVMRMixerBitmap9

            CDep_controllerDlg::BlendApplicationStepTwo();

   // Select the initial objects back into our device context
   DeleteObject(SelectObject(hdc, hbmOld));
   SelectObject(hdc, hOldFont);
   // Clean up resources
   DeleteObject(hbm);
   DeleteDC(hdc);

            //bmpInfo = NULL;
            //pBMP->Release();
            UpdateData(FALSE);
            hr = S_OK;
   return hr;
}

void CDep_controllerDlg::CaptureCurrentImage()
{
            BYTE* lpCurrImage = NULL;

            HRESULT hr;


            //Retrieve local date and time
            SYSTEMTIME sysCurrentTime;
            GetLocalTime(&sysCurrentTime);

            //Display information on screen
            m_sStatus2.Format("year %d, month %d, day %d hour
%d : %d",

            sysCurrentTime.wYear, sysCurrentTime.wMonth,
sysCurrentTime.wDay,

            sysCurrentTime.wHour, sysCurrentTime.wMinute);

            //setFileName including Chip No, Type of Cell, Date,
Time,
            CString tmpFileName, tmpFileTime,
tmpFileNameVideo, FileNameText;
            //Define Path
            tmpFileName = "c:\\thesis\\software\\data\\";

            tmpFileTime.Format("_%d_%d_%d_h%dm%d",

            sysCurrentTime.wYear, sysCurrentTime.wMonth,
sysCurrentTime.wDay,

            sysCurrentTime.wHour, sysCurrentTime.wMinute);

            sFileName = tmpFileName + CString("Chip_") +
m_sChipCode + "_" + m_sCellType + tmpFileTime;


            FileNameVideo =          sFileName + ".avi";

            FileNameText =           sFileName + ".dep";
            //File declarations and configuration
            //HANDLE hdlFile; //make it global //A - now is a global
variable
            LPCSTR FileName = FileNameText;
            //LPCSTR FileName =
"c:\\thesis\\software\\data\\texto1.dep";


            //Read the current video frame displayed into a byte??
buffer
            hr = pWindowlessControl-
>GetCurrentImage(&lpCurrImage);

            if (FAILED(hr))
  {
                        m_sStatus2 += "Couldn't get still image\n";
                        UpdateData(FALSE);
                        return;
  }

            //m_sStatus2 += "Getting still image\n";

            BITMAPFILEHEADER    hdr;
            DWORD               dwSize, dwWritten;
            LPBITMAPINFOHEADER  pdib =
(LPBITMAPINFOHEADER) lpCurrImage;


            //Create new file to store data
            CString sImageName, sTemp;
            sTemp.Format("_%.2fV_%.2fHz No.
%d",g_fVoltOut,g_fFreqOut, g_dPictureCounter);
            sImageName = sFileName + sTemp + ".bmp";

            HANDLE hFile = CreateFile((LPCTSTR)sImageName,
GENERIC_WRITE, FILE_SHARE_READ, NULL,

            CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, 0);

            if(hFile == INVALID_HANDLE_VALUE)
            {
                        m_sStatus2 += "Couldn't create file
name!!\n";
                        return;
            }

            m_sStatus2 += "Creating file name!!\n";

            // Initialize the bitmap header.
            dwSize = DibSize(pdib);
            hdr.bfType        = BFT_BITMAP;
            hdr.bfSize        = dwSize +
sizeof(BITMAPFILEHEADER);
            hdr.bfReserved1   = 0;
            hdr.bfReserved2   = 0;
            hdr.bfOffBits     =
(DWORD)sizeof(BITMAPFILEHEADER) + pdib->biSize +
            DibPaletteSize(pdib);

            // Write the bitmap header and bitmap bits to the file.
            WriteFile(hFile, (LPCVOID) &hdr,
sizeof(BITMAPFILEHEADER), &dwWritten, 0);
            WriteFile(hFile, (LPCVOID) pdib, dwSize, &dwWritten,
0);

            // Close the file.
            CloseHandle(hFile);

            // The app must free the image data returned from
GetCurrentImage()
            CoTaskMemFree(lpCurrImage);

            m_sStatus2 += "Captured current image to file!!!\n";
            //UpdateData(FALSE);
            g_dPictureCounter++;
            return;
}

void CDep_controllerDlg::CaptureVideo()
{
            HRESULT hr;

            //Attach pBuilder to Main Filter Graph
            pBuilder->SetFiltergraph(pGraph);


            ///////////////////////////////
            //Select Capture Device
            //Since I know that there is only one video input capture
device
            //it is only necesary to retrieve the first capture device of
the
            //Moniker List.  Should any other capture device was
connected to the
            //PC, this part of the code would be the first thing to be
modified
            //Refer to the DirectX Sample Browser, and select the
SysEnum executable
            ////////////////////////////////

            ICreateDevEnum *pDevEnum = NULL;
            IEnumMoniker *pEnum = NULL;

            // Create the System Device Enumerator.
            hr = CoCreateInstance(CLSID_SystemDeviceEnum,
NULL,

            CLSCTX_INPROC_SERVER, IID_ICreateDevEnum,

            reinterpret_cast<void**>(&pDevEnum));
            if (FAILED(hr))
            {
                        m_sStatus2 += "Error at creating system
device enumerator\n";
                        CDep_controllerDlg::CloseInterfaces();
                        return;
            }
            else
                        m_sStatus2 += "1.  creating system device
enumerator\n";

            //Creating System Device Enumerator
            hr = pDevEnum-
>CreateClassEnumerator(CLSID_VideoInputDeviceCategory,


            &pEnum, 0);
            if (FAILED(hr))
            {
                        m_sStatus2 += "Error at creating system
device enumerator\n";
                        pDevEnum->Release();
                        pDevEnum = NULL;
                        CDep_controllerDlg::CloseInterfaces();
                        return;
            }
            m_sStatus2 += "2.  Creating Class Enumerator\n";
```

```
        //Adding capture Filters
        IMoniker *pMoniker = NULL;

        //A while loop would be used at this part, but since we
knwo there is only
        //one capture filter we stop at the first iteration
        if(pEnum->Next(1, &pMoniker, NULL) == S_OK)
        {
                hr = pMoniker->BindToObject(0, 0,
IID_IBaseFilter, (void**)&pCapture);
                if (FAILED(hr))
                {
                        m_sStatus2 += "Error at
Binding Capture Filter\nNot Found\n";
                        pMoniker->Release();
                        pDevEnum->Release();
                        pEnum->Release();

        CDep_controllerDlg::CloseInterfaces();
                        return;
                }
                m_sStatus2 += "3.  binding to object\n";
                hr = pGraph->AddFilter(pCapture,
L"Capture Filter");

        }

        else
        {
                //pCapture->Release();
                pMoniker->Release();
                pDevEnum->Release();
                pDevEnum = NULL;
                pEnum->Release();
                m_sStatus2 += "Error at obtaining capture
filter\n";
                UpdateData(FALSE);
                CDep_controllerDlg::CloseInterfaces();
                return;  //add the releases for every COM
object or interface
        }

        //Query Information about the filter


        pMoniker->Release();
        pDevEnum->Release();
        pDevEnum = NULL;
        pEnum->Release();
        //The video card presents multiple input connectios, we
must choose
        //the composite input, which is pin 0, based on the Filter
Graph Tool
        IAMCrossbar *pXBar1 = NULL;
        hr = pBuilder-
>FindInterface(&LOOK_UPSTREAM_ONLY, NULL, pCapture,

                        IID_IAMCrossbar,
(void**)&pXBar1);
        if (FAILED(hr))
        {
                if(hr == E_NOINTERFACE)
                        m_sStatus2 += "No such
interface supported.\n";
                if(hr == E_POINTER)
                        m_sStatus2 += "NULL pointer
argument.\n";
```

```
                        m_sStatus2 += "Error at accesing crossbar
pXBar1\n";
                        UpdateData(FALSE);
                        CDep_controllerDlg::CloseInterfaces();
                        return;
                }
                m_sStatus2 += "4.  Getting Xbar\n";

                IBaseFilter *pFilter = NULL;
                hr = pXBar1->QueryInterface(IID_IBaseFilter,
(void**)&pFilter);
                if (FAILED(hr))
                {
                        pXBar1->Release();
                        m_sStatus2 += "Error at getting base
filter\n";
                        UpdateData(FALSE);
                        CDep_controllerDlg::CloseInterfaces();
                        return;
                }
                m_sStatus2 += "5.  Getting base filter\n";

                //Route Pins composite to output pin
                hr = pXBar1->Route(0,0);
                if (hr == S_FALSE)
                {
                        m_sStatus2 += "Cannot route these
pins\n";
                        pXBar1->Release();


                        UpdateData(FALSE);
                        CDep_controllerDlg::CloseInterfaces();
                        return;
                }
                m_sStatus2 += "6.  Routing Pins\n";

                pFilter->Release();
                pXBar1->Release();

                ////////////////////////////
                ////Recent
                ////////////////////////////
                //create VMR9 Filter
                hr = CoCreateInstance(CLSID_VideoMixingRenderer9,
0,
                                CLSCTX_INPROC_SERVER,
IID_IBaseFilter, (void**)&pVmr9);
                if (FAILED(hr))
   {
                        m_sStatus2 += "Error 2.  Creating VMR9
Filter\n";
                        UpdateData(FALSE);
                        CDep_controllerDlg::CloseInterfaces();
                        return;
                }
                m_sStatus2 += "7.  Creating VMR9 Filter\n";



                //Add Filter to Filter Graph Manager
                hr = pGraph->AddFilter(pVmr9, L"VMR9");
                if (FAILED(hr))
   {
                        m_sStatus2 += "Error 3.  Adding Filter in
VMR\n";
                        UpdateData(FALSE);
                        CDep_controllerDlg::CloseInterfaces();
                        return;
```

```
        }
        m_sStatus2 += "8.  Adding Filter in VMR\n";

        /////////////
        //This section was moved a little bit down
        ////////////
        ////////////////


        //Set Windowless Mode
        hr = CDep_controllerDlg::SetWindowlessMode();
        if (FAILED(hr))
  {
                m_sStatus2 += "Error 9.
SetWindowlessMode\n";
                UpdateData(FALSE);
                CDep_controllerDlg::CloseInterfaces();
                return;
        }
        m_sStatus2 += "11.  SetWindowlessMode\n";


        //////////////////////////////////
        //Render video
        //////////////////////////////////
        // Build the preview part of the graph.
        hr = pBuilder-
>RenderStream(&PIN_CATEGORY_PREVIEW,
&MEDIATYPE_Video,

                                pCapture, NULL, pVmr9);
//was NULL before
        if (FAILED(hr))
        {
                m_sStatus2 += "Failed on Render Stream
Preview\n";
                UpdateData(FALSE);
                CDep_controllerDlg::CloseInterfaces();
                return;
        }
        m_sStatus2 += "12.  Rendering stream in VMR\n";

        ///////////

        //Convert Filename in string format to BSTR
        USES_CONVERSION;
        LPCSTR psz3 = (LPCSTR) FileNameVideo;
        BSTR bstrFileName = A2BSTR(psz3);

        hr = pBuilder->SetOutputFileName(
                                &MEDIASUBTYPE_Avi,  //
 Specifies AVI for the target file.
                                bstrFileName,
                                //L"C:\\Example9.avi", // File
name.
                                &pMux,          // Receives a
pointer to the mux.
                                0);         // (Optional)
Receives a pointer to the file sink.


        if (FAILED(hr))
        {
                m_sStatus2 += "Failed on
SetOutputFileName\n";
                UpdateData(FALSE);
                CDep_controllerDlg::CloseInterfaces();
                return;
        }
        m_sStatus2 += "9.  set output file name in VMR\n";
```

```
        // Render the capture stream to the mux.

        hr = pBuilder-
>RenderStream(&PIN_CATEGORY_CAPTURE,
&MEDIATYPE_Video, pCapture,

                        NULL, pMux);
        if (FAILED(hr))
        {
                m_sStatus2 += "Failed on Render Stream
Capture\n";
                UpdateData(FALSE);
                CDep_controllerDlg::CloseInterfaces();
                return;
        }
        m_sStatus2 += "10.  render stream in VMR\n";


        ///////////

        //Set video position
        hr = CDep_controllerDlg::VideoPosition();
        if (FAILED(hr))
        {
                m_sStatus2 += "Failed on
VideoPosition\n";
                UpdateData(FALSE);
                CDep_controllerDlg::CloseInterfaces();
                return;
        }
        m_sStatus2 += "13.  VideoPosition\n";


        hr = CDep_controllerDlg::BlendApplicationImage();

        if (FAILED(hr))
  {
                CDep_controllerDlg::CloseInterfaces();
                m_sStatus2 += "14  Error
BlendApplicationImage\n";
                UpdateData(FALSE);
                return;
        }
        m_sStatus2 += "14.  BlendApplicationImage\n";


        //Start capturing video
        hr = pMediaControl->Run();
        if (FAILED(hr))
        {
                m_sStatus2 += "Failed on Running
Graph\n";
                UpdateData(FALSE);
                CDep_controllerDlg::CloseInterfaces();
                return;
        }
        else
                m_sStatus2 += "15. Media Control
Running\n";

        //Set Timer every 100 ms to capture current image
displayed
        gTimerInt = SetTimer(ID_TIMER_INT, 100, 0);

        UpdateData(FALSE);
}
```

```
void CDep_controllerDlg::CleanInterfaces()
{
        pVMR9FilterConfig = NULL;
        pWindowlessControl = NULL;
  pMediaControl = NULL;
  pMediaEvent = NULL;
  pGraph = NULL;
        pBuilder = NULL;
        pVmr9 = NULL;
        pMux = NULL;
}

void CDep_controllerDlg::CloseInterfaces()
{
        //Release the interface pointers and close the COM
library
        pWindowlessControl->Release();
        pVmr9->Release(); //

  pMediaControl->Release();
        pMediaEvent->SetNotifyWindow(NULL, 0, 0);
  pMediaEvent->Release();
  pGraph->Release();
        pBuilder->Release();
        CDep_controllerDlg::CleanInterfaces();
        CoUninitialize();
        m_sStatus2 += "Closing interfaces\n";
        UpdateData(FALSE);
}

HRESULT CDep_controllerDlg::GetInterfaces()
{
  HRESULT hr;

        //Initialize COM library
        hr = CoInitialize(NULL);
        if (FAILED(hr))
        {
                m_sStatus2 += "Failing at initialize COM
library\n";
                // Add error-handling code here.
                return hr;
        }

  // Create the filter graph
  hr = CoCreateInstance (CLSID_FilterGraph, NULL,
CLSCTX_INPROC_SERVER,
        IID_IGraphBuilder, (void **) &pGraph);
  if (FAILED(hr))
        {
                m_sStatus2 += "Failing at creating the filter
graph\n";
                // Add error-handling code here.
                return hr;
        }

  // Create the capture graph builder
  hr = CoCreateInstance (CLSID_CaptureGraphBuilder2 , NULL,
CLSCTX_INPROC_SERVER,
        IID_ICaptureGraphBuilder2, (void **) &pBuilder);
        if (FAILED(hr))
        {
                m_sStatus2 += "Failing at creating capture
filter graph\n";
                // Add error-handling code here.
                return hr;
        }

  // Obtain interfaces for media control and Video Window
```

```
  hr = pGraph->QueryInterface(IID_IMediaControl,(void
**)&pMediaControl);
        if (FAILED(hr))
        {
                m_sStatus2 += "Failing at obtaining
interface for Media Control\n";
                // Add error-handling code here.
                return hr;
        }

  hr = pGraph->QueryInterface(IID_IMediaEvent, (void
**)&pMediaEvent);
        if (FAILED(hr))
        {
                m_sStatus2 += "Failing at obtaining
interface for Media Event\n";
                // Add error-handling code here.
                return hr;
        }


        //Get the handle for the actual window
        g_hWnd = GetSafeHwnd();
  // Set the window handle used to process graph events
  hr = pMediaEvent->SetNotifyWindow((OAHWND)g_hWnd,
WM_GRAPHNOTIFY, 0);

  return hr;
}


void CDep_controllerDlg::HandleGraphEvent()
{
  LONG evCode, evParam1, evParam2;
  HRESULT hr;

  // Make sure that we don't access the media event interface
  // after it has already been released.
  if (!pMediaEvent)
        {
                m_sStatus2 += "No media event
interface\n";
                UpdateData(FALSE);
    return;
        }

        m_sStatus2 += "Within Handler\n";
        UpdateData(FALSE);


  // Process all queued events
  while(pMediaEvent->GetEvent(&evCode, (LONG_PTR *)
&evParam1,
        (LONG_PTR *) &evParam2, 0) == S_OK)
  {
    // Free memory associated with callback, since we're not using
it
    hr = pMediaEvent->FreeEventParams(evCode, evParam1,
evParam2);
                if (FAILED(hr))
                {
                        m_sStatus2 += "error freeing
event parameters\n";
                        UpdateData(FALSE);
                }

                //If the video has finalized stop the graph
                //still having an error when closing the
video capture window
```

```
                        if(evCode == (EC_COMPLETE ||
EC_USERABORT))
                        {
                                    KillTimer(gTimer);
                                    gTimer = 0;
                                    m_sStatus2 += "Finish
rendering\n";
                                    m_sStatus2 += "Stopping
video\n";
                                    hr = pMediaControl->Stop();
                                    pSource->Release();
                                    pPin->Release();
                                    pFilterGraph2->Release();

            CDep_controllerDlg::CloseInterfaces();
                                    UpdateData(FALSE);
                                    break;
                        }
            }
            return;
}

HRESULT CDep_controllerDlg::SetWindowlessMode()
{

            HRESULT hr;
            hr = pVmr9->QueryInterface(IID_IVMRFilterConfig9,
(void **)&pVMR9FilterConfig);
            if (FAILED(hr))
   {
                        m_sStatus2 += "Failed at getting
VMR9FilterConfig\n";
                        UpdateData(FALSE);
                        return hr;
   }

            m_sStatus2 += "Getting VMR9FilterConfig\n";
            //setting rendering mode to windowless
            hr = pVMR9FilterConfig-
>SetRenderingMode(VMR9Mode_Windowless);
            if (FAILED(hr))
   {

                        m_sStatus2 += "Failed at setting
windowless mode\n";
                        UpdateData(FALSE);
                        return hr;
   }
            pVMR9FilterConfig->SetNumberOfStreams(1);

            pVMR9FilterConfig->Release();
            m_sStatus2 += "Setting windowless mode\n";

            // Set the window.
            hr = pVmr9-
>QueryInterface(IID_IVMRWindowlessControl9,
(void**)&pWindowlessControl);
            if (FAILED(hr))
   {
                        m_sStatus2 += "Error at Associating
windowless control\n";
                        UpdateData(FALSE);
                        return hr;
   }
            m_sStatus2 += "Associating windowless control\n";

            //SetVideoClipping
            hr = pWindowlessControl-
>SetVideoClippingWindow(g_hWnd);
            if (FAILED(hr))
   {
                        m_sStatus2 += "Error at
SetVideoClippingWindow\n";
                        UpdateData(FALSE);
                        return hr;
   }
            m_sStatus2 += "SetVideoClippingWindow\n";

            //Set border color
            hr = pWindowlessControl-
>SetBorderColor(RGB(0,0,0));

#ifndef BILINEAR_FILTERING
        // Request point filtering (instead of bilinear filtering)
        // to improve the text quality.  In general, if you are
        // not scaling the app Image, you should use point filtering.
        // This is very important if you are doing source color keying.
        IVMRMixerControl9 *pMix;

        hr = pVmr9->QueryInterface(IID_IVMRMixerControl9,
(void**)&pMix);
        if( SUCCEEDED(hr))
        {
            DWORD dwPrefs=0;
            hr = pMix->GetMixingPrefs(&dwPrefs);

            if (SUCCEEDED(hr))
            {
                dwPrefs |= MixerPref_PointFiltering;
                dwPrefs &= ~(MixerPref_BiLinearFiltering);

                hr = pMix->SetMixingPrefs(dwPrefs);
            }
            pMix->Release();
        }
#endif

            m_sStatus2 += "SetVideoPosition\n";


            UpdateData(FALSE);
            return hr;
}

HRESULT CDep_controllerDlg::VideoPosition()
{
            // Find the native video size.
            long lWidth, lHeight, lARWidth,lARHeight;
            HRESULT hr;
            hr = pWindowlessControl-
>GetNativeVideoSize(&lWidth, &lHeight, &lARWidth, &lARHeight);
            if (FAILED(hr))
   {
                        m_sStatus2 += "Error at
GetNativeVideoSize\n";
                        UpdateData(FALSE);
                        return hr;
   }
            m_sStatus2 += "20. GetNativeVideoSize\n";

            CRect rcDest, rcSrc;

            GetClientRect(&rcDest);
            SetRect(&rcDest, 0, 0, rcDest.right, rcDest.bottom);
            SetRect(&rcSrc, 0, 0, lWidth, lHeight);

            CString sTemp;
            sTemp.Format("Windowless Mode Dest width -> %d,
heigth -> %d\nSource width -> %d height -> %d\n",
                                                rcDest.right,
rcDest.bottom, lWidth, lHeight);


            m_sStatus2 += sTemp;
            sTemp.Format("ratio width -> %d, heigth -> %d\n",
lARWidth,lARHeight);
            m_sStatus2 += sTemp;

            //We need to display the real size of the video, so we
do not need to scalate it
            hr = pWindowlessControl-
>SetVideoPosition(&rcSrc,&rcSrc);

            if (FAILED(hr))
   {
                        m_sStatus2 += "Error at
SetVideoPosition\n";
                        UpdateData(FALSE);
                        return hr;
   }
            m_sStatus2 += "SetVideoPosition in Video Position\n";


            UpdateData(FALSE);
            return hr;
}

void CDep_controllerDlg::OnTimer(UINT nIDEvent)
{
            // TODO: Add your message handler code here and/or
call default
            //Which timer triggered this event?
            CString sTemp;
            //float fAutFreqOut, fAutVoltOut;
            switch(nIDEvent)
            {
            case ID_TIMER_AUT:

                        //End of the timer cycle
                        if( gdCounterVolt > gdTotalAutVoltInt)
                        {
                                    //kill both timers
                                    KillTimer(ID_TIMER_AUT);
                                    KillTimer(ID_TIMER_INT);
                                    g_dFlagTimer = 1;
                        }


                        if(g_dFlagTimer!=1 && gdCounterFreq <=
gdTotalAutFreqInt)
                        {
                                    g_fFreqOut = g_fAutFreqMin +
(g_fAutFreqInt * gdCounterFreq);

                                    sTemp.Format("FREQ
%.2f\n",g_fFreqOut);
                                    sDataCommOut = (LPCSTR)
sTemp;

                                    //BlendApplication again to
display information

             CDep_controllerDlg::BlendApplicationAgain();


             CDep_controllerDlg::SendDataOutComm();

                                    gdCounterFreq++;
                                    //g_dFlagTimer = 0;
                        }
                        else
                        if( gdCounterFreq > gdTotalAutFreqInt &&
gdCounterVolt < gdTotalAutVoltInt && g_dFlagTimer !=1)
                        {
                                    //increment voltage and output
it
                                    gdCounterFreq = 0;
                                    gdCounterVolt++;
                                    g_fVoltOut = g_fAutVoltMin +
(g_fAutVoltInt * gdCounterVolt);
                                    //BlendApplication again to
display information


                                    g_fFreqOut = g_fAutFreqMin;
                                    sTemp.Format("FREQ
%.2f\n",g_fFreqOut);
                                    sDataCommOut = (LPCSTR)
sTemp;



                                    sTemp.Format("VOLT
%.2f\n",g_fVoltOut);
                                    sDataCommOut = (LPCSTR)
sTemp;

            CDep_controllerDlg::SendDataOutComm();



            CDep_controllerDlg::BlendApplicationAgain();
                        }



                        case ID_TIMER_INT:
                                    //commented on January 25,
2004 6:32
                                    if(m_sChipCode != "CAL")

            CDep_controllerDlg::CaptureCurrentImage();
                                    else
                                                break;

                        break;

                        //m_sStatus2.Format("Freq %.2f \n",
g_fFreqOut);
                        //m_sStatus2 += sTemp + "OnTimer\n";
                        UpdateData(FALSE);


            }
            CDialog::OnTimer(nIDEvent);
}

LRESULT CDep_controllerDlg::WindowProc(UINT message,
WPARAM wParam, LPARAM lParam)
{
            // TODO: Add your specialized code here and/or call
the base class
                        // TODO: Add your specialized code here
and/or call the base class
            //m_sStatus2 += "Entro al error handling\n";
            //UpdateData(FALSE);
            //An interruption has occurred
            switch (message)
   {
      case WM_GRAPHNOTIFY:
```

```
                                        m_sStatus2 += "Getting into
Window Proc\n";
                                        UpdateData(FALSE);

            CDep_controllerDlg::HandleGraphEvent();  //
Application-defined function.
        break;
      // Handle other Windows messages here too.
  }
            return CDialog::WindowProc(message, wParam,
lParam);
}

void CDep_controllerDlg::OnStopcapture()
{
            // TODO: Add your control notification handler code
here
            HRESULT hr;

            //commented on january 25 6:29
            KillTimer(ID_TIMER_INT);

            /////////////////////
            const DWORD wStartCookie = 1, wStopCookie = 2;

            hr = pBuilder-
>ControlStream(&PIN_CATEGORY_PREVIEW,
&MEDIATYPE_Video, pCapture,

                        0, NULL, wStartCookie, wStopCookie);

            if (FAILED(hr))
  {
                        m_sStatus2 += "Couldn't stop control
stream\n";
                        pVmr9->Release();
                        CDep_controllerDlg::CloseInterfaces();
                        return;
            }



/*
            hr = pMediaControl->Stop();
            if (FAILED(hr))
  {
                        m_sStatus2 += "Couldn't stop graph\n";
                        pVmr9->Release();
                        CDep_controllerDlg::CloseInterfaces();
                        return;
            }
*/

            m_sStatus2 += "Stop Capturing\n";
            UpdateData(FALSE);

            CDep_controllerDlg::CloseInterfaces();
}

void CDep_controllerDlg::OnPlayvideo()
{
            // TODO: Add your control notification handler code
here
            //VMR9 RENDER ALLOWS TO INCLUDE THE VIDEO
DISPLAY IN THE SAME APPLICATION
            //AS A WINDOWLESS MODE.  FOR FUTURE WORK
            HRESULT hr;

            gTimer = SetTimer(1, 100, 0);
```

```
            //Initialize COM libray and get interfaces
            m_sStatus2 += "1.  Getting interfaces\n";
            hr = CDep_controllerDlg::GetInterfaces();
            if (FAILED(hr))
  {
     m_sStatus2 += "Failed to get video interfaces!\n";
     return;
  }

            //create VMR9 Filter
            hr = CoCreateInstance(CLSID_VideoMixingRenderer9,
0,
                                          CLSCTX_INPROC_SERVER,
IID_IBaseFilter, (void**)&pVmr9);
            if (SUCCEEDED(hr))
  {
                        m_sStatus2 += "2.  Creating VMR9
Filter\n";
            }

            //Add Filter to Filter Graph Manager
            hr = pGraph->AddFilter(pVmr9, L"VMR9");
            if (SUCCEEDED(hr))
  {
                        m_sStatus2 += "3.  Adding Filter in
VMR\n";
            }


            //Add source filter
            hr = pGraph->AddSourceFilter(L"C:\\highway.avi",
L"Source1", &pSource);
            if (SUCCEEDED(hr))
  {
                        m_sStatus2 += "4.  Adding source Filter
...\n";
            }

            //Set windowless mode
            hr = CDep_controllerDlg::SetWindowlessMode();
            if (SUCCEEDED(hr))
  {
                        m_sStatus2 += "55.
SetWindowlessMode\n";
            }
            else
            {
                        CDep_controllerDlg::CloseInterfaces();
                        m_sStatus2 += "9.  Error trying to
SetWindowlessMode\n";
                        UpdateData(FALSE);
                        return;
            }

            hr = pSource->EnumPins(&pEnumPins);

            if (SUCCEEDED(hr))
  {
                        m_sStatus2 += "5.  Enumerating
Pins....\n";
            }

            //We know there is only one output pin, however we
            //have to obtain that pin and store in pPin
   while(pEnumPins->Next(1, &pPin, 0) == S_OK)
   {
      PIN_DIRECTION PinDirThis;
                        //query if the pin is either an input or output
pin
      hr = pPin->QueryDirection(&PinDirThis);
      if (FAILED(hr))
      {
         pPin->Release();
         pEnumPins->Release();
                                         m_sStatus2 += "Error
Enumerating Pins\n";

            CDep_controllerDlg::CloseInterfaces();
         return;

      }
                              m_sStatus2 += "6.  Getting pin\t";
                              if (PinDirThis == PINDIR_INPUT)
                                         m_sStatus2 +=
"PINDIR_INPUT\n";
                              if (PinDirThis == PINDIR_OUTPUT)
                                         m_sStatus2 +=
"PINDIR_OUTPUT\n";
      // Release the pin for the next time through the loop.
   }

                              pEnumPins->Release();

   hr = pGraph->QueryInterface(IID_IFilterGraph2, (void
**)&pFilterGraph2);
                  if (SUCCEEDED(hr))
                  {
                              m_sStatus2 += "7.  obtaining interface
FilterGraph2\n";
                              // Add error-handling code here.
                  }



                  //output pin is 0
                  hr = pFilterGraph2-
>RenderEx(pPin,AM_RENDEREX_RENDERTOEXISTINGRENDE
RERS, NULL);
                  if (SUCCEEDED(hr))
                  {
                              m_sStatus2 += "8.  Rendering Ex Pin\n";
                              // Add error-handling code here.
                  }

                  hr = CDep_controllerDlg::VideoPosition();

                  ////////////////////

                  hr = CDep_controllerDlg::BlendApplicationImage();

                  if (FAILED(hr))
   {
                              CDep_controllerDlg::CloseInterfaces();
                              m_sStatus2 += "10Error
 BlendApplicationImage\n";
                              UpdateData(FALSE);
                              return;
                  }
                  m_sStatus2 += "10.  BlendApplicationImage\n";

                              ////////////////
                  ShowWindow(SW_SHOWNORMAL);
                  UpdateWindow();
                  SetForegroundWindow();
                  SetFocus();

                  hr = pMediaControl->Run();
                  if (FAILED(hr))
   {
                              CDep_controllerDlg::CloseInterfaces();
```

```
                                         m_sStatus2 += "10Error trying to run
file\n";
                                         return;
                  }
                  m_sStatus2 += "10.  Running Graph\n";


                  m_sStatus2 += "11.  Succeded creating interfaces and
closing them up\n";
                  UpdateData(FALSE);
}

void CDep_controllerDlg::OnStill()
{
                  // TODO: Add your control notification handler code
here
                  CDep_controllerDlg::CaptureCurrentImage();
}

void CDep_controllerDlg::OnCapture()
{
                  // TODO: Add your control notification handler code
here
                  //Get Video Interfaces
                  HRESULT hr;
                  hr = CDep_controllerDlg::GetInterfaces();

                  if (FAILED(hr))
    {
       m_sStatus2 += "Failed to get video interfaces!";
       return;
    }

                  m_sStatus2 += "1.  Calling Capture Video Method!\n";
                  //Preview window on monitor;
                  UpdateData(FALSE);

                  CDep_controllerDlg::CaptureVideo();
}

void CDep_controllerDlg::OpenCommPort()
{
                  //Based on:
                  //http://msdn.microsoft.com/library/default.asp?url=/libra
ry/en-us/devio/base/configuring_a_communications_resource.asp

                  LPCSTR pcCommPort = "COM1";
                  m_sDataOutSerial = pcCommPort;
                  BOOL bSuccess;
                  sDataCommOut = "SYST:REM\n";
                  UpdateData(TRUE);

                  //Initialize instrument as Remote

                  //sSendData = m_sDataOurSerial + "\n";
                  //m_sStatus2 = sSendData + CString("\n");

                  hComm = CreateFile(pcCommPort,

                  GENERIC_READ|GENERIC_WRITE,

                  0, //exclusive acces

                  NULL, //no security attributes

                  OPEN_EXISTING, //always use OPEN_EXISTING

                  0, //not overlapped IO

                  NULL); //must be NULL  for COM devices
```

```
        if (hComm == INVALID_HANDLE_VALUE)
                m_sStatus = "Error trying to open COM1
port\n";
        else
                m_sStatus = "Oppening COM1 port\n";

        bSuccess = GetCommState(hComm, &dcb_comm);

        if(!bSuccess)
                m_sStatus += "Error trying to get COM1
state\n";
        else
                m_sStatus += "Retrieving COM1 settings
... \n";


        dcb_comm.BaudRate = CBR_9600;     // set the baud
rate
        dcb_comm.ByteSize = 8;            // data size, xmit, and
rcv
        dcb_comm.Parity = NOPARITY;       // no parity bit
        dcb_comm.StopBits = TWOSTOPBITS;  // one stop
bit

        // Build on the current configuration, and skip setting the
size
        // of the input and output buffers with SetupComm.
        //m_sStatus2 += sSendData;

        CDep_controllerDlg::SendDataOutComm();
        UpdateData(FALSE);
}

void CDep_controllerDlg::SendCommData(LPCSTR sData)
{
        //Not used anymore
}

void CDep_controllerDlg::SendDataOutComm()
{
        //Don't consider sData
        DWORD dDataLength=0, dTransmittedBytes;
        dDataLength = strlen(sDataCommOut);
        CString sLength;
        sLength.Format("Length %d", dDataLength);

        if(WriteFile(hComm, sDataCommOut, dDataLength,
&dTransmittedBytes, NULL) == TRUE)
        {
                m_sStatus += CString("Sending Data to
COM  ") + sDataCommOut + "\n";
                m_sStatus += sLength + "\n";
        }
        else
                m_sStatus2 += "Error writing to File\n";

        UpdateData(FALSE);

}

void CDep_controllerDlg::BlendApplicationStepTwo()
{
        HRESULT hr;
        // Get alpha-blended bitmap interface
   hr = pVmr9->QueryInterface(IID_IVMRMixerBitmap9,
(void**)&pBMP);

        if(FAILED(hr))
        {
```

```
                m_sStatus2 += "IID_IVMRMixerBitmap9
FAILED!\n";
                UpdateData(FALSE);
                return; //return hr
        }
        m_sStatus2 += "IID_IVMRMixerBitmap9
SUCCEDDED!\n";


   // Give the bitmap to the VMR for display
   hr = pBMP->SetAlphaBitmap(&bmpInfo);
   if (FAILED(hr))
        {
                m_sStatus2 += "SetAlphaBitmap
FAILED!\n";
                UpdateData(FALSE);
                return; //return hr
        }
        m_sStatus2 += "SetAlphaBitmap SUCCEDDED!\n";
}

void CDep_controllerDlg::BlendApplicationGetVideoSize()
{
   //LONG g_lVideoWidth, g_lVideoHeight;
   HRESULT hr;
   hr = pWindowlessControl->GetNativeVideoSize(&g_lVideoWidth,
&g_lVideoHeight, NULL, NULL);
   if (FAILED(hr))
        {
                m_sStatus2 += "Failed getting native
size\n";
      return; //return hr
        }

        CString sTemp;
        sTemp.Format("BB Dest width -> %d, heigth -> %d\n",
                                        &g_lVideoWidth,
g_lVideoHeight);
        m_sStatus2 += sTemp + "Getting Native Video Size\n";

        hFont = 0;

        UpdateData(FALSE);
}

void CDep_controllerDlg::BlendApplicationAgain()
{

        HRESULT hr;
        VMR9AlphaBitmap bmpInfoTemp = {0};
        hr = pBMP-
>GetAlphaBitmapParameters(&bmpInfoTemp);
   if (FAILED(hr))
        {
                m_sStatus2 += "Failed Getting Alpha
Bitmap Parameters!\n";
                UpdateData(FALSE);
                return; //return hr
        }
        m_sStatus2 += "GetAlphaBitmapParameters
SUCCEDDED!\n";

        //HRESULT hr;
        //Get Video Size

   // Create a device context compatible with the current window
        CClientDC hdc (this);
   HFONT hOldFont = (HFONT) SelectObject(hdc, hFont);

   // Determine the length of the string, then determine the
```

```
  // dimensions (in pixels) of the character string using the
  // currently selected font.  These dimensions are used to create
  // a bitmap below.
  int nTextBmpWidth, nTextBmpHeight;

            CString sDataTemp, sTest;
            sDataTemp.Format(" %.2f V %.2f Hz", g_fVoltOut,
g_fFreqOut);

            m_sStatus2 += "_____Data: " + sDataTemp +
"Blending Image____\n";
            // m_sChipCode + "V" "Hz"
            //sTest = "5 MHz 5 Volt DC chip1";
            sTest = "Chip " + m_sChipCode + sDataTemp;
            CSize zDimm;
            zDimm = hdc.GetTextExtent(sTest);
   nTextBmpHeight = zDimm.cy;
   nTextBmpWidth  = zDimm.cx;

   // Create a new bitmap that is compatible with the current window
            HBITMAP hbm = CreateCompatibleBitmap(hdc,
nTextBmpWidth, nTextBmpHeight);

   // Select our bitmap into the device context and save the old one
   BITMAP bm;
   HBITMAP hbmOld;
   GetObject(hbm, sizeof(bm), &bm);
   hbmOld = (HBITMAP)SelectObject(hdc, hbm);

   // Set initial bitmap settings
   RECT rcText;
   SetRect(&rcText, 0, 0, nTextBmpWidth, nTextBmpHeight);
   SetBkColor(hdc, RGB(255, 255, 255));          // Pure white
background
   SetTextColor(hdc, RGB(255, 0, 0));     // Write text with
requested color RED

            // Draw the requested text string onto the bitmap
            hdc.TextOut(0,0, sTest);


   // Configure the VMR's bitmap structure
   //VMR9AlphaBitmap bmpInfo; //made global
   ZeroMemory(&bmpInfoTemp, sizeof(bmpInfoTemp) );
   bmpInfoTemp.dwFlags = VMR9AlphaBitmap_hDC;
   bmpInfoTemp.hdc = hdc;
                                        // DC which has selected our
bitmap

   // Remember the width of this new bitmap
   dImageWidth = bm.bmWidth;

   // Save the ratio of the bitmap's width to the width of the video file.
   // This value is used to reposition the bitmap in composition
space.
   fBitmapCompWidth = (float)dImageWidth / (float)g_lVideoWidth;

   // Display the bitmap in the bottom right corner.
   // rSrc specifies the source rectangle in the GDI device context
   // rDest specifies the destination rectangle in composition space
(0.0f to 1.0f)
   bmpInfoTemp.rDest.left  = 0.0f + X_EDGE_BUFFER;
   bmpInfoTemp.rDest.right = 1.0f - X_EDGE_BUFFER;
   bmpInfoTemp.rDest.top = (float)(g_lVideoHeight - bm.bmHeight) /
(float)g_lVideoHeight - Y_EDGE_BUFFER;
   bmpInfoTemp.rDest.bottom = 1.0f - Y_EDGE_BUFFER;


   // Transparency value 1.0 is opaque, 0.0 is transparent.
            bmpInfoTemp.fAlpha = 0.8f;
```

```
            // Set the COLORREF so that the bitmap outline will be
transparent
   bmpInfoTemp.clrSrcKey = RGB(255, 255, 255);  // Pure white
   bmpInfoTemp.dwFlags |= VMRBITMAP_SRCCOLORKEY;

            // put text on screen
            bmpInfoTemp.rSrc = rcText;


            //Blend Image second step, calling Query Interface and
IVMRMixerBitmap9

   // Give the bitmap to the VMR for display
   hr = pBMP->UpdateAlphaBitmapParameters(&bmpInfoTemp);
   if (FAILED(hr))
            {
                        m_sStatus2 += "Failed Update Alpha
Bitmap Parameters!\n";
                        UpdateData(FALSE);
                        return; //return hr
            }
            m_sStatus2 += "SetAlphaBitmap AGAIN
SUCCEDDED!\n";


            // Give the bitmap to the VMR for display
   hr = pBMP->SetAlphaBitmap(&bmpInfoTemp);
   if (FAILED(hr))
            {
                        m_sStatus2 += "SetAlphaBitmap
FAILED!\n";
                        UpdateData(FALSE);
                        return; //return hr
            }
            m_sStatus2 += "SetAlphaBitmap SUCCEDDED!\n";



   // Select the initial objects back into our device context
   DeleteObject(SelectObject(hdc, hbmOld));
   SelectObject(hdc, hOldFont);
   // Clean up resources
   DeleteObject(hbm);
   DeleteDC(hdc);

            //bmpInfo = NULL;
            //pBMP->Release();
            UpdateData(FALSE);

}

void CDep_controllerDlg::OnRecord()
{
/*
            // TODO: Add your control notification handler code
here
            HRESULT hr;

            USES_CONVERSION;
            LPCSTR psz3 = (LPCSTR) FileNameVideo;
            BSTR bstrFileName = A2BSTR(psz3);


            hr = pBuilder->SetOutputFileName(
                                    &MEDIASUBTYPE_Avi,  //
Specifies AVI for the target file.
                                    bstrFileName,
                                    //L"C:\\Example9.avi", // File
name.
                                    &pMux,          // Receives a
pointer to the mux.
```

```
                                        0);             // (Optional)
Receives a pointer to the file sink.


        if (FAILED(hr))
        {
                m_sStatus2 += "Failed on
SetOutputFileName\n";
                UpdateData(FALSE);
                CDep_controllerDlg::CloseInterfaces();
                return;
        }
        m_sStatus2 += "9.  set output file name in VMR\n";


        // Render the capture stream to the mux.

        hr = pBuilder-
>RenderStream(&PIN_CATEGORY_CAPTURE,
&MEDIATYPE_Video, pCapture,

                NULL, pMux);
        if (FAILED(hr))
        {
                m_sStatus2 += "Failed on Render Stream
Capture\n";
                UpdateData(FALSE);
                CDep_controllerDlg::CloseInterfaces();
                return;
        }
        m_sStatus2 += "10.  render stream in VMR\n";

                //Start capturing video
        hr = pMediaControl->Run();
```

```
        if (FAILED(hr))
        {
                m_sStatus2 += "Failed on Running
Graph\n";
                UpdateData(FALSE);
                CDep_controllerDlg::CloseInterfaces();
                return;
        }
        else
                m_sStatus2 += "15. Media Control
Running\n";
        UpdateData(FALSE);
*/
}

void CDep_controllerDlg::setVideoOutputFormat()
{
        //set video frame and resolution
        // not completed
        //Set Video Output Format
        HRESULT hr;
        IAMStreamConfig *pConfig = NULL;
        hr = pBuilder->FindInterface(

        &PIN_CATEGORY_PREVIEW, // Preview pin.

        0,   // Any media type.

        pCapture, // Pointer to the capture filter.

        IID_IAMStreamConfig, (void**)&pConfig);

}
```

File Name: **dep_controller.cpp**

```
// dep_controller.cpp : Defines the class behaviors for the
application.
//

#include "stdafx.h"
#include "dep_controller.h"
#include "dep_controllerDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CDep_controllerApp

BEGIN_MESSAGE_MAP(CDep_controllerApp, CWinApp)
	//{{AFX_MSG_MAP(CDep_controllerApp)
			// NOTE - the ClassWizard will add and
remove mapping macros here.
			//    DO NOT EDIT what you see in these
blocks of generated code!
	//}}AFX_MSG
	ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CDep_controllerApp construction

CDep_controllerApp::CDep_controllerApp()
{
	// TODO: add construction code here,
	// Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CDep_controllerApp object

CDep_controllerApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CDep_controllerApp initialization

BOOL CDep_controllerApp::InitInstance()
{
	AfxEnableControlContainer();

	// Standard initialization
	// If you are not using these features and wish to reduce
the size
	// of your final executable, you should remove from the
following
	// the specific initialization routines you do not need.

#ifdef _AFXDLL
	Enable3dControls();			// Call
this when using MFC in a shared DLL
#else
	Enable3dControlsStatic();		// Call this when
linking to MFC statically
#endif

	CDep_controllerDlg dlg;
	m_pMainWnd = &dlg;
	int nResponse = dlg.DoModal();
	if (nResponse == IDOK)
	{
		// TODO: Place code here to handle when
the dialog is
		//  dismissed with OK
	}
	else if (nResponse == IDCANCEL)
	{
		// TODO: Place code here to handle when
the dialog is
		//  dismissed with Cancel
	}

	// Since the dialog has been closed, return FALSE so
that we exit the
	//  application, rather than start the application's
message pump.
	return FALSE;
}
```

File Name: **dep_controller.h**

```
// dep_controller.h : main header file for the DEP_CONTROLLER
application
//

#if
!defined(AFX_DEP_CONTROLLER_H__ACB218F0_FD0E_4D6A_
86AD_9C80B5588434__INCLUDED_)
#define
AFX_DEP_CONTROLLER_H__ACB218F0_FD0E_4D6A_86AD_9
C80B5588434__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#ifndef __AFXWIN_H__
	#error include 'stdafx.h' before including this file for
PCH
#endif

#include "resource.h"		// main symbols

/////////////////////////////////////////////////////////////////////////////
// CDep_controllerApp:
// See dep_controller.cpp for the implementation of this class
//

class CDep_controllerApp : public CWinApp
{
public:
	CDep_controllerApp();

// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CDep_controllerApp)
	public:
	virtual BOOL InitInstance();
	//}}AFX_VIRTUAL

// Implementation

	//{{AFX_MSG(CDep_controllerApp)
		// NOTE - the ClassWizard will add and
remove member functions here.
		//    DO NOT EDIT what you see in these
blocks of generated code !
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};


/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.

#endif //
!defined(AFX_DEP_CONTROLLER_H__ACB218F0_FD0E_4D6A_
86AD_9C80B5588434__INCLUDED_)
```

File Name: **dep_controllerDlg.h**

```
// dep_controllerDlg.h : header file
//

#if \
!defined(AFX_DEP_CONTROLLERDLG_H__B7583037_5ED8_43
A7_AF30_315E997054E6__INCLUDED_)
#define
AFX_DEP_CONTROLLERDLG_H__B7583037_5ED8_43A7_AF30
_315E997054E6__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////////
// CDep_controllerDlg dialog

class CDep_controllerDlg : public CDialog
{
// Construction
public:
		int gTimerInt;
		int g_dPictureCounter;
		int g_dFlagTimer;
		float g_fAutVoltInt;
		float g_fAutVoltMax;
		float g_fAutVoltMin;
		float g_fAutFreqMax;
		float g_fAutFreqMin;
		float g_fAutFreqInt;
		void setVideoOutputFormat();
		CString sFileName;
		void BlendApplicationAgain();
		LONG g_lVideoHeight;
		LONG g_lVideoWidth;
		void BlendApplicationGetVideoSize();
		void BlendApplicationStepTwo();
		VMR9AlphaBitmap bmpInfo;
		float g_fVoltOut;
		float g_fFreqOut;
		CString FileNameVideo;
		int gdCounterVolt;
		int gdCounterFreq;
		int gdTotalAutVoltInt;
		int gdTotalAutFreqInt;
		float gfTotalAutFreqInt;
		float gfPeriodAutFreqInt;
		float gfAutFreqOut;
		int gCountIntTimerAut;
		void SendDataOutComm();
		LPCSTR sDataCommOut;
		void SendCommData(LPCSTR sData);
		CString m_sDataOutSerial;
		HANDLE hComm;
		DCB dcb_comm;
		void OpenCommPort();
		HRESULT VideoPosition();
		HRESULT SetWindowlessMode();
		void HandleGraphEvent();
		HRESULT GetInterfaces();
		void CloseInterfaces();
		void CleanInterfaces();
		void CaptureVideo();
		void CaptureCurrentImage();
		HFONT hFont;
		int dImageWidth;
		float fBitmapCompWidth;
		HWND g_hWnd;
		int gTimer;
		IVMRMixerBitmap9 * pBMP;
		ICaptureGraphBuilder2 * pBuilder;
		IBaseFilter * pCapture;
		IEnumPins * pEnumPins;
		IFilterGraph2 * pFilterGraph2;
		IGraphBuilder * pGraph;
		IMediaControl * pMediaControl;
		IMediaEventEx * pMediaEvent;
		IBaseFilter * pMux;
		IPin * pPin;
		IBaseFilter * pSource;
		IBaseFilter * pVmr9;
		IVMRFilterConfig9 * pVMR9FilterConfig;
		IVMRWindowlessControl9 * pWindowlessControl;
		HRESULT BlendApplicationImage();
		void SwapRealVal(float *Value, int Selection);
		int dFlag;
		void Create_File();
		HANDLE hdlFileDep;
		CDep_controllerDlg(CWnd* pParent = NULL);	//
standard constructor

// Dialog Data
		//{{AFX_DATA(CDep_controllerDlg)
		enum { IDD = IDD_DEP_CONTROLLER_DIALOG };
		CButton	m_ctlAutStop;
		CComboBox		m_cmbManFreqSel;
		CComboBox		m_cmbGenWaveform;
		CComboBox		m_cmbGenTermination;
		CComboBox		m_cmbAutFreqSelMin;
		CComboBox		m_cmbAutFreqSelMax;
		CComboBox		m_cmbAutFreqSelInt;
		float	m_fCellCond;
		float	m_fCellPerm;
		CString	m_sCellPrep;
		float	m_fCellSize;
		CString	m_sCellSurf;
		CString	m_sCellType;
		CString	m_sChipCode;
		CString	m_sCommentsChip;
		CString	m_sCommentsGral;
		float	m_fMediumCond;
		CString	m_sMediumPrep;
		CString	m_sMediumReagent;
		CString	m_sStatus;
		CString	m_sStatus2;
		float	m_fAutFreqMax;
		float	m_fAutFreqMin;
		float	m_fAutVoltInt;
		float	m_fAutVoltMax;
		float	m_fAutVoltMin;
		float	m_fGenOffset;
		float	m_fManFreq;
		float	m_fManVolt;
		float	m_fMediumHumd;
		float	m_fMediumPH;
		float	m_fMediumTemp;
		float	m_fAutFreqInt;
		int			m_dAutTimerInt;
		//}}AFX_DATA

		// ClassWizard generated virtual function overrides
		//{{AFX_VIRTUAL(CDep_controllerDlg)
		protected:
		virtual void DoDataExchange(CDataExchange* pDX);
		// DDX/DDV support
```

```
	virtual LRESULT WindowProc(UINT message,
WPARAM wParam, LPARAM lParam);
	//}}AFX_VIRTUAL

// Implementation
protected:
	HICON m_hIcon;

	// Generated message map functions
	//{{AFX_MSG(CDep_controllerDlg)
	virtual BOOL OnInitDialog();
	afx_msg void OnSysCommand(UINT nID, LPARAM
lParam);
	afx_msg void OnPaint();
	afx_msg HCURSOR OnQueryDragIcon();
	afx_msg void OnStoreData();
	afx_msg void OnClearData();
	afx_msg void OnStoreComments();
	virtual void OnCancel();
	afx_msg void OnClearComments();
	afx_msg void OnNewData();
	afx_msg void OnAutApply();
	afx_msg void OnAutStop();
	afx_msg void OnAutRes();
	afx_msg void OnManApply();
	afx_msg void OnTimer(UINT nIDEvent);
	afx_msg void OnStopcapture();
	afx_msg void OnPlayvideo();
	afx_msg void OnStill();
	afx_msg void OnCapture();
	afx_msg void OnRecord();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
private:
	HANDLE hdlFile;
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.

#endif //
!defined(AFX_DEP_CONTROLLERDLG_H__B7583037_5ED8_43
A7_AF30_315E997054E6__INCLUDED_)
```

File Name: **Resources.h**

```
// dep_controllerDlg.h : header file
//

#if
!defined(AFX_DEP_CONTROLLERDLG_H__B7583037_5ED8_43
A7_AF30_315E997054E6__INCLUDED_)
#define
AFX_DEP_CONTROLLERDLG_H__B7583037_5ED8_43A7_AF30
_315E997054E6__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////////
// CDep_controllerDlg dialog

class CDep_controllerDlg : public CDialog
{
// Construction
public:
            int gTimerInt;
            int g_dPictureCounter;
            int g_dFlagTimer;
            float g_fAutVoltInt;
            float g_fAutVoltMax;
            float g_fAutVoltMin;
            float g_fAutFreqMax;
            float g_fAutFreqMin;
            float g_fAutFreqInt;
            void setVideoOutputFormat();
            CString sFileName;
            void BlendApplicationAgain();
            LONG g_lVideoHeight;
            LONG g_lVideoWidth;
            void BlendApplicationGetVideoSize();
            void BlendApplicationStepTwo();
            VMR9AlphaBitmap bmpInfo;
            float g_fVoltOut;
            float g_fFreqOut;
            CString FileNameVideo;
            int gdCounterVolt;
            int gdCounterFreq;
            int gdTotalAutVoltInt;
            int gdTotalAutFreqInt;
            float gfTotalAutFreqInt;
            float gfPeriodAutFreqInt;
            float gfAutFreqOut;
            int gCountIntTimerAut;
            void SendDataOutComm();
            LPCSTR sDataCommOut;
            void SendCommData(LPCSTR sData);
            CString m_sDataOutSerial;
            HANDLE hComm;
            DCB dcb_comm;
            void OpenCommPort();
            HRESULT VideoPosition();
            HRESULT SetWindowlessMode();
            void HandleGraphEvent();
            HRESULT GetInterfaces();
            void CloseInterfaces();
            void CleanInterfaces();
            void CaptureVideo();
            void CaptureCurrentImage();
            HFONT hFont;
            int dImageWidth;
            float fBitmapCompWidth;
            HWND g_hWnd;
            int gTimer;
            IVMRMixerBitmap9 * pBMP;
            ICaptureGraphBuilder2 * pBuilder;
            IBaseFilter * pCapture;
            IEnumPins * pEnumPins;
            IFilterGraph2 * pFilterGraph2;
            IGraphBuilder * pGraph;
            IMediaControl * pMediaControl;
            IMediaEventEx * pMediaEvent;
            IBaseFilter * pMux;
            IPin * pPin;
            IBaseFilter * pSource;
            IBaseFilter * pVmr9;
            IVMRFilterConfig9 * pVMR9FilterConfig;
            IVMRWindowlessControl9 * pWindowlessControl;
            HRESULT BlendApplicationImage();
            void SwapRealVal(float *Value, int Selection);
            int dFlag;
            void Create_File();
            HANDLE hdlFileDep;
            CDep_controllerDlg(CWnd* pParent = NULL);     //
standard constructor

// Dialog Data
            //{{AFX_DATA(CDep_controllerDlg)
            enum { IDD = IDD_DEP_CONTROLLER_DIALOG };
            CButton     m_ctlAutStop;
            CComboBox           m_cmbManFreqSel;
            CComboBox           m_cmbGenWaveform;
            CComboBox           m_cmbGenTermination;
            CComboBox           m_cmbAutFreqSelMin;
            CComboBox           m_cmbAutFreqSelMax;
            CComboBox           m_cmbAutFreqSelInt;
            float       m_fCellCond;
            float       m_fCellPerm;
            CString     m_sCellPrep;
            float       m_fCellSize;
            CString     m_sCellSurf;
            CString     m_sCellType;
            CString     m_sChipCode;
            CString     m_sCommentsChip;
            CString     m_sCommentsGral;
            float       m_fMediumCond;
            CString     m_sMediumPrep;
            CString     m_sMediumReagent;
            CString     m_sStatus;
            CString     m_sStatus2;
            float       m_fAutFreqMax;
            float       m_fAutFreqMin;
            float       m_fAutVoltInt;
            float       m_fAutVoltMax;
            float       m_fAutVoltMin;
            float       m_fGenOffset;
            float       m_fManFreq;
            float       m_fManVolt;
            float       m_fMediumHumd;
            float       m_fMediumPH;
            float       m_fMediumTemp;
            float       m_fAutFreqInt;
            int                 m_dAutTimerInt;
            //}}AFX_DATA

            // ClassWizard generated virtual function overrides
            //{{AFX_VIRTUAL(CDep_controllerDlg)
            protected:
            virtual void DoDataExchange(CDataExchange* pDX);
            // DDX/DDV support
            virtual LRESULT WindowProc(UINT message,
WPARAM wParam, LPARAM lParam);
            //}}AFX_VIRTUAL
// Implementation
protected:
        HICON m_hIcon;

        // Generated message map functions
        //{{AFX_MSG(CDep_controllerDlg)
        virtual BOOL OnInitDialog();
        afx_msg void OnSysCommand(UINT nID, LPARAM
lParam);
        afx_msg void OnPaint();
        afx_msg HCURSOR OnQueryDragIcon();
        afx_msg void OnStoreData();
        afx_msg void OnClearData();
        afx_msg void OnStoreComments();
        virtual void OnCancel();
        afx_msg void OnClearComments();
        afx_msg void OnNewData();
        afx_msg void OnAutApply();
        afx_msg void OnAutStop();
        afx_msg void OnAutRes();
        afx_msg void OnManApply();
        afx_msg void OnTimer(UINT nIDEvent);
        afx_msg void OnStopcapture();
        afx_msg void OnPlayvideo();
        afx_msg void OnStill();
        afx_msg void OnCapture();
        afx_msg void OnRecord();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
private:
        HANDLE hdlFile;
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.

#endif //
!defined(AFX_DEP_CONTROLLERDLG_H__B7583037_5ED8_43
A7_AF30_315E997054E6__INCLUDED_)
```

File Name: **StdAfx.cpp**

```
// stdafx.cpp : source file that includes just the standard includes
//              dep_controller.pch will be the pre-compiled header
//              stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"
```

File Name: **StdAfx.h**

```
// stdafx.h : include file for standard system include files,
//  or project specific include files that are used frequently, but
//      are changed infrequently
//

#if !defined(AFX_STDAFX_H__FF139901_ACAB_4A00_B9EA_2A0C3B94BD4C__INCLUDED_)
#define AFX_STDAFX_H__FF139901_ACAB_4A00_B9EA_2A0C3B94BD4C__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define VC_EXTRALEAN                        // Exclude rarely-used stuff from Windows headers

#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions
#include <afxdisp.h>        // MFC Automation classes
#include <afxdtctl.h>                       // MFC support for Internet Explorer 4 Common Controls
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>                         // MFC support for Windows Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT

////////////////////////////////////////////
//Author Jose Garcia
////////////////////////////////////////////
//Direct show library should never be left out
#include <Dshow.h>
#include <D3d9.h>
#include <Vmr9.h>
#include <afxpriv.h>
//////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_STDAFX_H__FF139901_ACAB_4A00_B9EA_2A0C3B94BD4C__INCLUDED_)
```

Appendix G

Software Code for the Control of the Solenoid Valves

This software application permits the control of the solenoid valves through the parallel port of a computer.

```
#include "stdafx.h"
#include "conio.h"
#include "stdio.h"
#include "time."
#define PORTADDRESS 0x378
#define DATA PORTADDRESS+0
#define STATUS PORTADDRESS+1
#define CONTROL PORTADDRESS+2


void wait(int seconds)
{

clock_t endwait;
endwait = clock () + seconds * CLK_TCK;
while (clock() < endwait) {}
}

int main()
{

// Activate pump

_outp(CONTROL, _inp(CONTROL) & 0xDF);
/* Forward Direction */

// Operation of the pump three valves

// - - V1 V2 V3 - - -

//0 0 1 0 1 0 0 0
_outp(DATA, 0x28);
wait(1);

//0 0 1 0 0 0 0 0
_outp(DATA, 0x20);
wait(1);

//0 0 1 1 0 0 0 0
_outp(DATA, 0x30);
wait(1);

//0 0 0 1 0 0 0 0
_outp(DATA, 0x10);
wait(1);
```

```
//0 0 0 1 1 0 0 0
_outp(DATA, 0x18);
wait(1);

//0 0 0 0 1 0 0 0
_outp(DATA, 0x08);
wait(1);


valve 1 closed
//0 1 0 0 0 0 0 0
_outp(DATA, 0x40);
wait(1);

valve 1 open
//0 0 0 0 0 0 0 0
_outp(DATA, 0x00);
wait(1);

valve 2 closed
//0 0 0 0 0 1 0 0
_outp(DATA, 0x04);
wait(1);

valve 2 open
//0 0 0 0 0 0 0 0
_outp(DATA, 0x00);
wait(1);

inlet port closed
//1 0 0 0 0 0 0 0
_outp(DATA, 0x80);
wait(1);

inlet port open
//0 0 0 0 0 0 0 0
_outp(DATA, 0x00);

printf("Test ended");
return 0;
}
```

Appendix H

Matlab Code Created by FEMLAB

Matlab code generated by FEMLAB in the simulation of the electric field distribution for the polynomial electrode configuration.

```
FEMLAB Model M-file
% Generated by FEMLAB 3.0a (FEMLAB 3.0.0.228, $Date:
2004/04/05 18:04:31 $)

flclear xfem

% Femlab version
clear vrsn
vrsn.name = 'FEMLAB 3.0';
vrsn.ext = 'a';
vrsn.major = 0;
vrsn.build = 228;
vrsn.rcs = '$Name:  $';
vrsn.date = '$Date: 2004/04/05 18:04:31 $';
xfem.version = vrsn;

% Geometry 2
g1=rect2(1,1.2,'base','corner','pos',[-0.4,-1]);
g2=rect2('125','125','base','corner','pos',{'0','0'},'rot','0');
carr={curve2([75,125],[0,50],[1,1]), ...
  curve2([125,125],[50,0],[1,1]), ...
  curve2([125,75],[0,0],[1,1])};
g3=geomcoerce('solid',carr);
g4=geomcomp({g2,g3},'ns',{'g2','g3'},'sf','g2-g3','edge','none');
garr=geomarrayr(g4,175,0,4,1);
[g5,g6,g7,g8]=deal(garr{:});
g6=rotate(g6,4.71238898038469,[0,0]);
g6=move(g6,[165.00000000000003,300]);
g6=move(g6,[10.000000000000028,0]);
g7=rotate(g7,1.5707963267948966,[0,0]);
g7=move(g7,[475,-350]);
g8=rotate(g8,3.141592653589793,[0,0]);
g8=move(g8,[825,-35.00000000000007]);
g7=move(g7,[-355,-165]);
g7=move(g7,[5,0]);
g7=move(g7,[0,-10]);
g8=move(g8,[0,-15]);
g4=move(g4,[0,0]);
g6=move(g6,[0,0]);
g7=move(g7,[0,0]);
g8=move(g8,[0,0]);
g4=move(g4,[0,175]);
g6=move(g6,[0,175]);
g7=move(g7,[0,175]);
g8=move(g8,[0,175]);
carr={curve2([0,-75],[0,-75],[1,1]), ...
  curve2([-75,-35],[-75,-75],[1,1]), ...
  curve2([-35,40],[-75,0],[1,1]), ...
  curve2([40,0],[0,0],[1,1])};
g9=geomcoerce('solid',carr);
carr={curve2([-75,-75],[-75,-35],[1,1]), ...
  curve2([-75,0],[-35,40],[1,1]), ...
  curve2([0,0],[40,0],[1,1]), ...
  curve2([0,-75],[0,-75],[1,1])};
g10=geomcoerce('solid',carr);
carr={curve2([300,375],[0,-75],[1,1]), ...
  curve2([375,375],[-75,-35],[1,1]), ...
  curve2([375,300],[-35,40],[1,1]), ...
  curve2([300,300],[40,0],[1,1])};
g11=geomcoerce('solid',carr);
carr={curve2([300,260],[0,0],[1,1]), ...
  curve2([260,335],[0,-75],[1,1]), ...
  curve2([335,375],[-75,-75],[1,1]), ...
  curve2([375,300],[-75,0],[1,1])};
g12=geomcoerce('solid',carr);
carr={curve2([0,-75],[300,375],[1,1]), ...
  curve2([-75,-75],[375,335],[1,1]), ...
  curve2([-75,0],[335,260],[1,1]), ...
  curve2([0,0],[260,300],[1,1])};
g13=geomcoerce('solid',carr);
carr={curve2([-75,-35],[375,375],[1,1]), ...
  curve2([-35,40],[375,300],[1,1]), ...
  curve2([40,0],[300,300],[1,1]), ...
  curve2([0,-75],[300,375],[1,1])};
g14=geomcoerce('solid',carr);
carr={curve2([300,360],[300,355],[1,1]), ...
  curve2([360,365],[355,335],[1,1]), ...
  curve2([365,300],[335,300],[1,1])};
g15=geomcoerce('solid',carr);
clear g15
carr={curve2([300,375],[300,375],[1,1]), ...
  curve2([375,375],[375,335],[1,1]), ...
  curve2([375,300],[335,260],[1,1]), ...
  curve2([300,300],[260,300],[1,1])};
g16=geomcoerce('solid',carr);
carr={curve2([300,260],[300,300],[1,1]), ...
  curve2([260,335],[300,375],[1,1]), ...
  curve2([335,375],[375,375],[1,1]), ...
  curve2([375,300],[375,300],[1,1])};
g17=geomcoerce('solid',carr);
g18=extrude(g4,'Distance',5,'Scale',[1;1],'Displ',[0;0],'Twist',0,'
Wrkpln',[0 1 0;0 0 1;0 0 0]);
g19=extrude(g6,'Distance',5,'Scale',[1;1],'Displ',[0;0],'Twist',0,'
Wrkpln',[0 1 0;0 0 1;0 0 0]);
g20=extrude(g7,'Distance',5,'Scale',[1;1],'Displ',[0;0],'Twist',0,'
Wrkpln',[0 1 0;0 0 1;0 0 0]);
g21=extrude(g8,'Distance',5,'Scale',[1;1],'Displ',[0;0],'Twist',0,'
Wrkpln',[0 1 0;0 0 1;0 0 0]);
g22=extrude(g9,'Distance',5,'Scale',[1;1],'Displ',[0;0],'Twist',0,'
Wrkpln',[0 1 0;0 0 1;0 0 0]);
g23=extrude(g10,'Distance',5,'Scale',[1;1],'Displ',[0;0],'Twist',0
,'Wrkpln',[0 1 0;0 0 1;0 0 0]);
```

```
g24=extrude(g11,'Distance',5,'Scale',[1;1],'Displ',[0;0],'Twist',0
,'Wrkpln',[0 1 0;0 0 1;0 0 0]);
g25=extrude(g12,'Distance',5,'Scale',[1;1],'Displ',[0;0],'Twist',0
,'Wrkpln',[0 1 0;0 0 1;0 0 0]);
g26=extrude(g13,'Distance',5,'Scale',[1;1],'Displ',[0;0],'Twist',0
,'Wrkpln',[0 1 0;0 0 1;0 0 0]);
g27=extrude(g14,'Distance',5,'Scale',[1;1],'Displ',[0;0],'Twist',0
,'Wrkpln',[0 1 0;0 0 1;0 0 0]);
g28=extrude(g16,'Distance',5,'Scale',[1;1],'Displ',[0;0],'Twist',0
,'Wrkpln',[0 1 0;0 0 1;0 0 0]);
g29=extrude(g17,'Distance',5,'Scale',[1;1],'Displ',[0;0],'Twist',0
,'Wrkpln',[0 1 0;0 0 1;0 0 0]);

% Geometry 1
g30=geomcomp({g19,g28,g29},'ns',{'EXT2','EXT11','EXT12'},'
sf','EXT2+EXT11+EXT12','face','all','edge','all');
g31=geomcomp({g18,g26,g27},'ns',{'g18','g26','g27'},'sf','g18+
g26+g27','face','all','edge','all');
g32=geomcomp({g20,g22,g23},'ns',{'g20','g22','g23'},'sf','g20+
g22+g23','face','all','edge','all');
g33=geomcomp({g21,g24,g25},'ns',{'g21','g24','g25'},'sf','g21+
g24+g25','face','all','edge','all');
g30=move(g30,[75,75,0]);
g31=move(g31,[75,75,0]);
g32=move(g32,[75,75,0]);
g33=move(g33,[75,75,0]);
g34=block3('450','450','50','base','corner','pos',{'0','0','0'},'axis',
{'0','0','1'},'rot','0');
g35=geomcomp({g30,g31,g32,g33,g34},'ns',{'CO1','CO2','CO
3','CO4','BLK1'},'sf','BLK1-CO1-CO2-CO3-
CO4','face','all','edge','all');
flclear fem
clear s
s.objs={g35};
s.name={'CO5'};
s.tags={'g35'};

fem.draw=struct('s',s);
fem.geom=geomcsg(fem);

% Constants
xfem.const={'V1','5','V2','-5'};

% Initialize mesh for geometry 1
fem.mesh=meshinit(fem);

% Refine mesh for geometry 1
fem.mesh=meshrefine(fem, ...
            'rmethod','longest');
xfem.fem{1}=fem;

% (Default values are not included)

fem=xfem.fem{1};

% Application mode 1
clear appl
appl.mode.class = 'ElectrostaticsGeneralized';
appl.assignsuffix = '_qv';
clear bnd
bnd.V0 = {0,0,'V1','V2'};
bnd.type = {'V0','n0','V','V'};
bnd.ind =
[2,2,1,1,2,1,2,1,1,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,3,3,3,3,4,4,4.
4,3,4,3,3,4,4,3,3,3,4,4,4,2];
appl.bnd = bnd;
fem.appl{1} = appl;
xfem.fem{1} = fem;

flclear fem
```

```
fem.sdim = {'x','y'};
xfem.fem{2} = fem;

% Multiphysics
xfem=multiphysics(xfem);

% Extend mesh
xfem.xmesh=meshextend(xfem,'geoms',[1]);

% Solve problem
xfem.sol=femlin(xfem, ...
          'symmetric','on', ...
          'solcomp',{'V'}, ...
          'outcomp',{'V'}, ...
          'nonlin','off', ...
          'linsolver','gmres', ...
          'prefun','amg');

% Save current fem structure for restart purposes
fem0=xfem;

% Plot solution
postplot(xfem, ...
      'slicedata',{'V','cont','internal'}, ...
      'slicexspacing',5, ...
      'sliceyspacing',0, ...
      'slicezspacing',0, ...
      'slicemap','jet(1024)', ...
      'title','Slice: Electric potential', ...
      'refine',1, ...
      'grid','on', ...
      'campos',[-1457.7188911867422,-
1967.9619543538129,1620.893166850452], ...
      'transparency',0.40000000000000013, ...
      'renderer','opengl');

% Plot solution
postplot(xfem, ...
      'slicedata',{'normE_qv','cont','internal'}, ...
      'slicexspacing',0, ...
      'sliceyspacing',0, ...
      'slicezspacing',1, ...
      'slicemap','jet(1024)', ...
      'title','Slice: Electric field, norm', ...
      'refine',1, ...
      'grid','on', ...
      'campos',[-1457.7188911867422,-
1967.9619543538129,1620.893166850452], ...
      'transparency',0.40000000000000013, ...
      'renderer','opengl');

% Geometry 1
g35=scale(g35,1E-6,1E-6,1E-6,0,0,0);
fem=xfem.fem{1};
clear s
s.objs={g35};
s.name={'CO5'};
s.tags={'g35'};

fem.draw=struct('s',s);
fem.geom=geomcsg(fem);

% Initialize mesh for geometry 1
fem.mesh=meshinit(fem);

% Initialize mesh for geometry 1
fem.mesh=meshinit(fem);

% Initialize mesh for geometry 1
fem.mesh=meshinit(fem, ...
            'hmaxfact',0.8, ...
            'hcutoff',0.008, ...
            'hgrad',1.35, ...
            'hcurve',0.35);
xfem.fem{1}=fem;

% (Default values are not included)

fem=xfem.fem{1};

% Application mode 1
clear appl
appl.mode.class = 'ElectrostaticsGeneralized';
appl.assignsuffix = '_qv';
clear bnd
bnd.V0 = {0,0,'V1','V2'};
bnd.type = {'n0','V0','V','V'};
bnd.ind =
[1,1,2,2,1,2,1,2,2,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3,4,4,4,
4,3,4,3,3,4,4,3,3,3,4,4,4,1];
appl.bnd = bnd;
fem.appl{1} = appl;
xfem.fem{1} = fem;

fem=xfem.fem{2};
fem.sdim = {'x','y'};
xfem.fem{2} = fem;

% Multiphysics
xfem=multiphysics(xfem);

% Extend mesh
xfem.xmesh=meshextend(xfem,'geoms',[1]);

% Evaluate initial value using current solution
init = asseminit(xfem,'u',fem0.sol,'xmesh',fem0.xmesh);

% Solve problem
xfem.sol=femlin(xfem, ...
            'init',init, ...
            'symmetric','on', ...
            'solcomp',{'V'}, ...
            'outcomp',{'V'}, ...
            'nonlin','off', ...
            'linsolver','gmres', ...
            'prefun','amg');

% Save current fem structure for restart purposes
fem0=xfem;

% Plot solution
postplot(xfem, ...
      'slicedata',{'normE_qv','cont','internal'}, ...
      'slicexspacing',0, ...
      'sliceyspacing',0, ...
      'slicezspacing',1, ...
      'slicemap','jet(1024)', ...
      'title','Slice: Electric field, norm', ...
      'refine',2, ...
      'grid','on', ...
      'campos',[-2.4731763724039645E-4,-
0.001764881717946996,0.0024754665806867864], ...
      'transparency',0.40000000000000013, ...
      'renderer','opengl');

% Plot solution
postplot(xfem, ...
      'slicedata',{'Ex_qv','cont','internal'}, ...
      'slicexspacing',0, ...
      'sliceyspacing',0, ...
      'slicezspacing',1, ...
      'slicemap','jet(1024)', ...
      'title','Slice: Electric field, x component', ...
      'refine',2, ...
      'grid','on', ...
      'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
      'transparency',0.7000000000000001, ...
      'renderer','opengl');

% Plot solution
postplot(xfem, ...
      'slicedata',{'Ey_qv','cont','internal'}, ...
      'slicexspacing',0, ...
      'sliceyspacing',0, ...
      'slicezspacing',1, ...
      'slicemap','jet(1024)', ...
      'title','Slice: Electric field, y component', ...
      'refine',2, ...
      'grid','on', ...
      'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
      'transparency',0.7000000000000001, ...
      'renderer','opengl');

% Plot solution
postplot(xfem, ...
      'isodata',{'normE_qv','cont','internal'}, ...
      'isolevels',5, ...
      'isomap','jet(1024)', ...
      'isostriptol',0.01, ...
      'title','Isosurface: Electric field, norm', ...
      'refine',2, ...
      'grid','on', ...
      'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
      'transparency',0.7000000000000001, ...
      'renderer','opengl');

% Plot solution
postplot(xfem, ...
      'isodata',{'normE_qv','cont','internal'}, ...
      'isolevels',5, ...
      'isomap','jet(1024)', ...
      'isostriptol',0.01, ...
      'title','Isosurface: Electric field, norm', ...
      'refine',2, ...
      'grid','on', ...
      'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
      'transparency',0.7000000000000001, ...
      'renderer','opengl');

% Plot solution
postplot(xfem, ...
      'isodata',{'normE_qv','cont','internal'}, ...
      'isolevels',5, ...
      'isomap','jet(1024)', ...
      'isostriptol',0.01, ...
      'arrowdata',{'Ex_qv','Ey_qv','Ez_qv'}, ...
      'arrowxspacing',7, ...
      'arrowyspacing',7, ...
      'arrowzspacing',7, ...
      'arrowtype','cone', ...
      'arrowstyle','proportional', ...
      'arrowcolor',[1.0,0.0,0.0], ...
      'title','Isosurface: Electric field, norm   Arrow: Electric
field', ...
      'refine',2, ...
      'grid','on', ...
      'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
      'transparency',0.7000000000000001, ...
      'renderer','opengl');

% Plot solution
postplot(xfem, ...
      'isodata',{'normE_qv','cont','internal'}, ...
      'isolevels',5, ...
      'isomap','jet(1024)', ...
      'isostriptol',0.01, ...
      'arrowdata',{'Ex_qv','Ey_qv','Ez_qv'}, ...
      'arrowxspacing',5, ...
      'arrowyspacing',5, ...
      'arrowzspacing',5, ...
      'arrowtype','cone', ...
      'arrowstyle','proportional', ...
      'arrowcolor',[1.0,0.0,0.0], ...
      'title','Isosurface: Electric field, norm   Arrow: Electric
field', ...
      'refine',2, ...
      'grid','on', ...
      'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
      'transparency',0.7000000000000001, ...
      'renderer','opengl');

% Plot solution
postplot(xfem, ...
      'isodata',{'normE_qv','cont','internal'}, ...
      'isolevels',5, ...
      'isomap','jet(1024)', ...
      'isostriptol',0.01, ...
      'arrowdata',{'Ex_qv','Ey_qv','Ez_qv'}, ...
      'arrowxspacing',20, ...
      'arrowyspacing',20, ...
      'arrowzspacing',20, ...
      'arrowtype','cone', ...
      'arrowstyle','proportional', ...
      'arrowcolor',[1.0,0.0,0.0], ...
      'title','Isosurface: Electric field, norm   Arrow: Electric
field', ...
      'refine',2, ...
      'grid','on', ...
      'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
      'transparency',0.7000000000000001, ...
      'renderer','opengl');
fem=xfem.fem{1};

% Initialize mesh for geometry 1
fem.mesh=meshinit(fem, ...
            'hmaxfact',0.8, ...
            'hcutoff',0.008, ...
            'hgrad',1.35, ...
            'hcurve',0.35);
xfem.fem{1}=fem;

% (Default values are not included)

fem=xfem.fem{1};

% Application mode 1
clear appl
appl.mode.class = 'ElectrostaticsGeneralized';
appl.assignsuffix = '_qv';
clear bnd
bnd.sigmabnd = {0,0,0,100};
bnd.V0 = {0,'V1','V2',0};
bnd.type = {'n0','V','V','ss'};
bnd.ind =
[1,1,4,4,1,4,1,4,4,1,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,2,2,2,2,3,3,3,
3,2,3,2,2,3,3,2,2,2,3,3,3,1];
appl.bnd = bnd;
fem.appl{1} = appl;
xfem.fem{1} = fem;

fem=xfem.fem{2};
fem.sdim = {'x','y'};
xfem.fem{2} = fem;

% Multiphysics
xfem=multiphysics(xfem);

% Extend mesh
xfem.xmesh=meshextend(xfem,'geoms',[1]);

% Evaluate initial value using current solution
init = asseminit(xfem,'u',fem0.sol,'xmesh',fem0.xmesh);

% Solve problem
xfem.sol=femlin(xfem, ...
            'init',init, ...
            'symmetric','on', ...
            'solcomp',{'V'}, ...
            'outcomp',{'V'}, ...
            'nonlin','off', ...
            'linsolver','gmres', ...
            'prefun','amg');

% Save current fem structure for restart purposes
fem0=xfem;

% Plot solution
postplot(xfem, ...
      'isodata',{'normE_qv','cont','internal'}, ...
      'isolevels',5, ...
      'isomap','jet(1024)', ...
      'isostriptol',0.01, ...
      'arrowdata',{'Ex_qv','Ey_qv','Ez_qv'}, ...
      'arrowxspacing',20, ...
      'arrowyspacing',20, ...
      'arrowzspacing',20, ...
      'arrowtype','cone', ...
      'arrowstyle','proportional', ...
      'arrowcolor',[1.0,0.0,0.0], ...
      'title','Isosurface: Electric field, norm   Arrow: Electric
field', ...
      'refine',2, ...
      'grid','on', ...
      'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
      'transparency',0.7000000000000001, ...
      'renderer','opengl');

% Plot solution
postplot(xfem, ...
      'slicedata',{'normE_qv','cont','internal'}, ...
      'slicexspacing',0, ...
      'sliceyspacing',0, ...
      'slicezspacing',1, ...
      'slicemap','jet(1024)', ...
      'title','Slice: Electric field, norm', ...
      'refine',2, ...
      'grid','on', ...
```

```
        'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
        'transparency',0.7000000000000001, ...
        'renderer','opengl');

% Plot solution
postplot(xfem, ...
        'slicedata',{'normE_qv','cont','internal'}, ...
        'slicexspacing',0, ...
        'sliceyspacing',0, ...
        'slicezspacing',[5E-6], ...
        'slicemap','jet(1024)', ...
        'title','Slice: Electric field, norm', ...
        'refine',2, ...
        'grid','on', ...
        'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
        'transparency',0.7000000000000001, ...
        'renderer','opengl');

% Plot solution
postplot(xfem, ...
        'slicedata',{'normE_qv','cont','internal'}, ...
        'slicexspacing',0, ...
        'sliceyspacing',0, ...
        'slicezspacing',[8E-6], ...
        'slicemap','jet(1024)', ...
        'title','Slice: Electric field, norm', ...
        'refine',2, ...
        'grid','on', ...
        'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
        'transparency',0.7000000000000001, ...
        'renderer','opengl');

% Plot solution
postplot(xfem, ...
        'slicedata',{'normE_qv','cont','internal'}, ...
        'slicexspacing',0, ...
        'sliceyspacing',0, ...
        'slicezspacing',[8E-6], ...
        'slicemap','hot(1024)', ...
        'title','Slice: Electric field, norm', ...
        'refine',2, ...
        'grid','on', ...
        'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
        'transparency',0.7000000000000001, ...
        'renderer','opengl');

% Plot solution
postplot(xfem, ...
        'slicedata',{'normE_qv','cont','internal'}, ...
        'slicexspacing',0, ...
        'sliceyspacing',0, ...
        'slicezspacing',[8E-6], ...
        'slicemap','hot(512)', ...
        'title','Slice: Electric field, norm', ...
        'refine',2, ...
        'grid','on', ...
        'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
        'transparency',0.7000000000000001, ...
        'renderer','opengl');

% Plot solution
postplot(xfem, ...
        'slicedata',{'normE_qv','cont','internal'}, ...
        'slicexspacing',0, ...
        'sliceyspacing',0, ...
        'slicezspacing',[10E-6], ...
        'slicemap','hot(512)', ...
        'title','Slice: Electric field, norm', ...
        'refine',2, ...
        'grid','on', ...
        'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
        'transparency',0.7000000000000001, ...
        'renderer','opengl');

% Plot solution
postplot(xfem, ...
        'slicedata',{'normE_qv','cont','internal'}, ...
        'slicexspacing',0, ...
        'sliceyspacing',0, ...
        'slicezspacing',[5E-6], ...
        'slicemap','hot(512)', ...
        'title','Slice: Electric field, norm', ...
        'refine',2, ...
        'grid','on', ...
        'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
        'transparency',0.7000000000000001, ...
        'renderer','opengl');

% Plot solution
postplot(xfem, ...
        'slicedata',{'normE_qv','cont','internal'}, ...
        'slicexspacing',0, ...
        'sliceyspacing',0, ...
        'slicezspacing',[6E-6], ...
        'slicemap','hot(512)', ...
        'title','Slice: Electric field, norm', ...
        'refine',2, ...
        'grid','on', ...
        'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
        'transparency',0.7000000000000001, ...
        'renderer','opengl');

% Plot solution
postplot(xfem, ...
        'slicedata',{'normE_qv','cont','internal'}, ...
        'slicexspacing',0, ...
        'sliceyspacing',0, ...
        'slicezspacing',[7E-6], ...
        'slicemap','hot(512)', ...
        'title','Slice: Electric field, norm', ...
        'refine',2, ...
        'grid','on', ...
        'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
        'transparency',0.7000000000000001, ...
        'renderer','opengl');

% Plot solution
postplot(xfem, ...
        'slicedata',{'normE_qv','cont','internal'}, ...
        'slicexspacing',0, ...
        'sliceyspacing',0, ...
        'slicezspacing',[7E-6], ...
        'slicemap','cool(512)', ...
        'title','Slice: Electric field, norm', ...
        'refine',2, ...
        'grid','on', ...
        'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
        'transparency',0.7000000000000001, ...
        'renderer','opengl');

% Plot solution
postplot(xfem, ...
        'slicedata',{'normE_qv','cont','internal'}, ...
        'slicexspacing',0, ...
        'sliceyspacing',0, ...
        'slicezspacing',[7E-6], ...
        'slicemap','gray(512)', ...
        'title','Slice: Electric field, norm', ...
        'refine',2, ...
        'grid','on', ...
        'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
        'transparency',0.7000000000000001, ...
        'renderer','opengl');

% Plot solution
postplot(xfem, ...
        'slicedata',{'normE_qv','cont','internal'}, ...
        'slicexspacing',0, ...
        'sliceyspacing',0, ...
        'slicezspacing',[7E-6], ...
        'slicemap','bone(512)', ...
        'title','Slice: Electric field, norm', ...
        'refine',2, ...
        'grid','on', ...
        'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
        'transparency',0.7000000000000001, ...
        'renderer','opengl');

% Plot solution
postplot(xfem, ...
        'slicedata',{'normE_qv','cont','internal'}, ...
        'slicexspacing',0, ...
        'sliceyspacing',0, ...
        'slicezspacing',[7E-6], ...
        'slicemap','pink(512)', ...
        'title','Slice: Electric field, norm', ...
        'refine',2, ...
        'grid','on', ...
        'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
        'transparency',0.7000000000000001, ...
        'renderer','opengl');

% Plot solution
postplot(xfem, ...
        'slicedata',{'normE_qv','cont','internal'}, ...
        'slicexspacing',0, ...
        'sliceyspacing',0, ...
        'slicezspacing',[7E-6], ...
        'slicemap','pink(1024)', ...
        'title','Slice: Electric field, norm', ...
        'refine',2, ...
        'grid','on', ...
        'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
        'transparency',0.7000000000000001, ...
        'renderer','opengl');

% Plot solution
postplot(xfem, ...
        'slicedata',{'normE_qv','cont','internal'}, ...
        'slicexspacing',0, ...
        'sliceyspacing',0, ...
        'slicezspacing',[7E-6], ...
        'slicemap','jet(1024)', ...
        'title','Slice: Electric field, norm', ...
        'refine',2, ...
        'grid','on', ...
        'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
        'transparency',0.7000000000000001, ...
        'renderer','opengl');

% Plot solution
postplot(xfem, ...
        'slicedata',{'normE_qv','cont','internal'}, ...
        'slicexspacing',0, ...
        'sliceyspacing',0, ...
        'slicezspacing',[7E-6], ...
        'slicemap','jet(2048)', ...
        'title','Slice: Electric field, norm', ...
        'refine',2, ...
        'grid','on', ...
        'campos',[2.2499999613501132E-
4,1.3240731304096752E-4,0.0032167862780834323], ...
        'transparency',0.7000000000000001, ...
        'renderer','opengl');

% Plot solution
postplot(xfem, ...
        'slicedata',{'normE_qv','cont','internal'}, ...
        'slicedlim',[1.164825e5 4.44505e5], ...
        'slicexspacing',0, ...
        'sliceyspacing',0, ...
        'slicezspacing',[7E-6], ...
        'slicemap','jet(2048)', ...
        'title','Slice: Electric field, norm', ...
        'refine',2, ...
        'grid','on', ...
        'campos',[2.2499999613501132E-
4,1.227300766176536E-4,0.0032167862780834323], ...
        'transparency',0.7000000000000001, ...
        'renderer','opengl');

% Plot solution
postplot(xfem, ...
        'slicedata',{'normE_qv','cont','internal'}, ...
        'slicedlim',[1.164825e5 4.44505e4], ...
        'slicexspacing',0, ...
        'sliceyspacing',0, ...
        'slicezspacing',[7E-6], ...
        'slicemap','jet(2048)', ...
        'title','Slice: Electric field, norm', ...
        'refine',2, ...
        'grid','on', ...
        'campos',[2.2499999613501132E-
4,1.227300766176536E-4,0.0032167862780834323], ...
        'transparency',0.7000000000000001, ...
        'renderer','opengl');

% Plot solution
postplot(xfem, ...
        'slicedata',{'normE_qv','cont','internal'}, ...
        'slicedlim',[1.164825e5 4.44505e3], ...
        'slicexspacing',0, ...
        'sliceyspacing',0, ...
        'slicezspacing',[7E-6], ...
        'slicemap','jet(2048)', ...
        'title','Slice: Electric field, norm', ...
        'refine',2, ...
        'grid','on', ...
        'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
```

```
    'transparency',0.7000000000000001, ...
    'renderer','opengl');

% Plot solution
postplot(xfem, ...
    'slicedata',{'normE_qv','cont','internal'}, ...
    'slicedlim',[1.164825e5 4.44505e3], ...
    'slicexspacing',0, ...
    'sliceyspacing',0, ...
    'slicezspacing',[7E-6], ...
    'slicemap','jet(1024)', ...
    'title','Slice: Electric field, norm', ...
    'refine',2, ...
    'grid','on', ...
    'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
    'transparency',0.7000000000000001, ...
    'renderer','opengl');

% Plot solution
postplot(xfem, ...
    'slicedata','normE_qv', ...
    'slicedlim',[1.164825e5 4.44505e3], ...
    'slicexspacing',0, ...
    'sliceyspacing',0, ...
    'slicezspacing',[7E-6], ...
    'slicemap','jet(1024)', ...
    'title','Slice: Electric field, norm', ...
    'refine',2, ...
    'grid','on', ...
    'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
    'transparency',0.7000000000000001, ...
    'renderer','opengl');

% Plot solution
postplot(xfem, ...
    'slicedata','normE_qv', ...
    'slicexspacing',0, ...
    'sliceyspacing',0, ...
    'slicezspacing',[7E-6], ...
    'slicemap','jet(1024)', ...
    'title','Slice: Electric field, norm', ...
    'refine',2, ...
    'grid','on', ...
    'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
    'transparency',0.7000000000000001, ...
    'renderer','opengl');

% Plot solution
postplot(xfem, ...
    'slicedata','normE_qv', ...
    'slicedlim',[1.473686e3 4.531616e5], ...
    'slicexspacing',0, ...
    'sliceyspacing',0, ...
    'slicezspacing',[7E-6], ...
    'slicemap','jet(1024)', ...
    'title','Slice: Electric field, norm', ...
    'refine',2, ...
    'grid','on', ...
    'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
    'transparency',0.7000000000000001, ...
    'renderer','opengl');

% Plot solution
postplot(xfem, ...
    'slicedata','normE_qv', ...
    'slicedlim',[1.473686e4 4.531616e5], ...
    'slicexspacing',0, ...
    'sliceyspacing',0, ...
    'slicezspacing',[7E-6], ...
    'slicemap','jet(1024)', ...
    'title','Slice: Electric field, norm', ...
    'refine',2, ...
    'grid','on', ...
    'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
    'transparency',0.7000000000000001, ...
    'renderer','opengl');

% Plot solution
postplot(xfem, ...
    'slicedata','normE_qv', ...
    'slicedlim',[5e4 4.531616e5], ...
    'slicexspacing',0, ...
    'sliceyspacing',0, ...
    'slicezspacing',[7E-6], ...
    'slicemap','jet(1024)', ...
    'title','Slice: Electric field, norm', ...
    'refine',2, ...
    'grid','on', ...
    'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
    'transparency',0.7000000000000001, ...
    'renderer','opengl');

% Plot solution
postplot(xfem, ...
    'slicedata','normE_qv', ...
    'slicedlim',[5e3 4.531616e5], ...
    'slicexspacing',0, ...
    'sliceyspacing',0, ...
    'slicezspacing',[7E-6], ...
    'slicemap','jet(1024)', ...
    'title','Slice: Electric field, norm', ...
    'refine',2, ...
    'grid','on', ...
    'campos',[2.2499999613501132E-
4,2.2499999613501132E-4,0.0032167862780834323], ...
    'transparency',0.7000000000000001, ...
    'renderer','opengl');

% Geometry 2
fem=xfem.fem{2};
clear s
s.objs={g4,g6,g7,g8,g9,g10,g11,g12,g13,g14,g16,g17};
s.name={'CO2','CO1','CO3','CO4','CO5','CO6','CO7','CO8','C
O9','CO10','CO11','CO12'};
s.tags={'g4','g6','g7','g8','g9','g10','g11','g12','g13','g14','g16','g
17'};

fem.draw=struct('s',s);
xfem.fem{2}=fem;
% FEMLAB Model M-file
% Generated by FEMLAB 3.0a (FEMLAB 3.0.0.228, $Date:
2004/04/05 18:04:31 $)

% Plot solution
postplot(xfem, ...
    'slicedata','normE_qv', ...
    'slicedlim',[5e3 4.531616e5], ...
    'slicexspacing',0, ...
    'sliceyspacing',0, ...
    'slicezspacing',1, ...
```

```
        'slicemap','jet(1024)', ...
        'title','Slice: Electric field, norm', ...
        'refine',2, ...
        'grid','on', ...
        'campos',[-0.0014577188660629965,-
0.0019679619204399173,0.0016208931387259396], ...
        'transparency',0.7000000000000001, ...
        'renderer','opengl');

% Plot solution
postplot(xfem, ...
        'slicedata','normE_qv', ...
        'slicexspacing',0, ...
        'sliceyspacing',0, ...
        'slicezspacing',1, ...
        'slicemap','jet(1024)', ...
        'title','Slice: Electric field, norm', ...
        'refine',2, ...
        'grid','on', ...
        'campos',[-0.0014577188660629965,-
0.0019679619204399173,0.0016208931387259396], ...
        'transparency',0.7000000000000001, ...
        'renderer','opengl');

% Plot solution
postplot(xfem, ...
        'slicedata','normE_qv', ...
        'slicexspacing',0, ...
        'sliceyspacing',0, ...
        'slicezspacing',1, ...
        'slicemap','jet(1024)', ...
        'title','Slice: Electric field, norm', ...
        'refine',2, ...
        'grid','on', ...
        'campos',[-0.0014577188660629965,-
0.0019679619204399173,0.0016208931387259396], ...
        'transparency',0.7000000000000001, ...
        'renderer','opengl');

% Plot solution
postplot(xfem, ...
        'slicedata','normE_qv', ...
        'slicexspacing',0, ...
        'sliceyspacing',0, ...
        'slicezspacing',1, ...
        'slicemap','jet(1024)', ...
        'tetdata',{'normE_qv','cont','internal'}, ...
        'tetmap','jet(1024)', ...
        'tetkeep',1, ...
        'tetkeeptype','random', ...
        'title','Slice: Electric field, norm   Tetrahedron: Electric
field, norm', ...
        'refine',2, ...
        'grid','on', ...
        'campos',[-0.0014577188660629965,-
0.0019679619204399173,0.0016208931387259396], ...
        'transparency',0.7000000000000001, ...
        'renderer','opengl');

% Plot solution
postplot(xfem, ...
        'slicedata','normE_qv', ...
        'slicexspacing',0, ...
        'sliceyspacing',0, ...
        'slicezspacing',1, ...
        'slicemap','jet(1024)', ...
        'isodata',{'normE_qv','cont','internal'}, ...
        'isolevels',5, ...
        'isomap','jet(1024)', ...
        'isostriptol',0.01, ...
        'title','Slice: Electric field, norm   Isosurface: Electric
field, norm', ...
        'refine',2, ...
        'grid','on', ...
        'campos',[-0.0014577188660629965,-
0.0019679619204399173,0.0016208931387259396], ...
        'transparency',0.7000000000000001, ...
        'renderer','opengl');

% Geometry 2
fem=xfem.fem{2};
clear s
s.objs={g17,g4,g6,g10,g16,g14,g9,g12,g7,g8,g11,g13};
s.name={'CO12','CO2','CO1','CO6','CO11','CO10','CO5','CO8'
,'CO3','CO4','CO7','CO9'};
s.tags={'g17','g4','g6','g10','g16','g14','g9','g12','g7','g8','g11','g
13'};

fem.draw=struct('s',s);
xfem.fem{2}=fem;
```

Appendix I

List of Supplementary Material

Different files such as the code generated by FEMLAB, schematics, printed circuit board layouts, the End Note Reference file, and other supplementary material were stored in a CD. The specific location of all of them is given below.

End Note Reference file:
\Thesis\Ref_End_Note\

Electrode and PDMS mask designs:
\Thesis\LEdit\Designs

Schematics and PCB layouts (not updated):
\Thesis\Hardware\Schematics\

FEMLAB output files:
\Thesis\Simulations\FEMLAB\

Software code for the Graphical User Interface:
\Thesis\Software\Dep_Controller\

A soft copy of the papers listed in the Bibliography:
\Thesis\Literature\